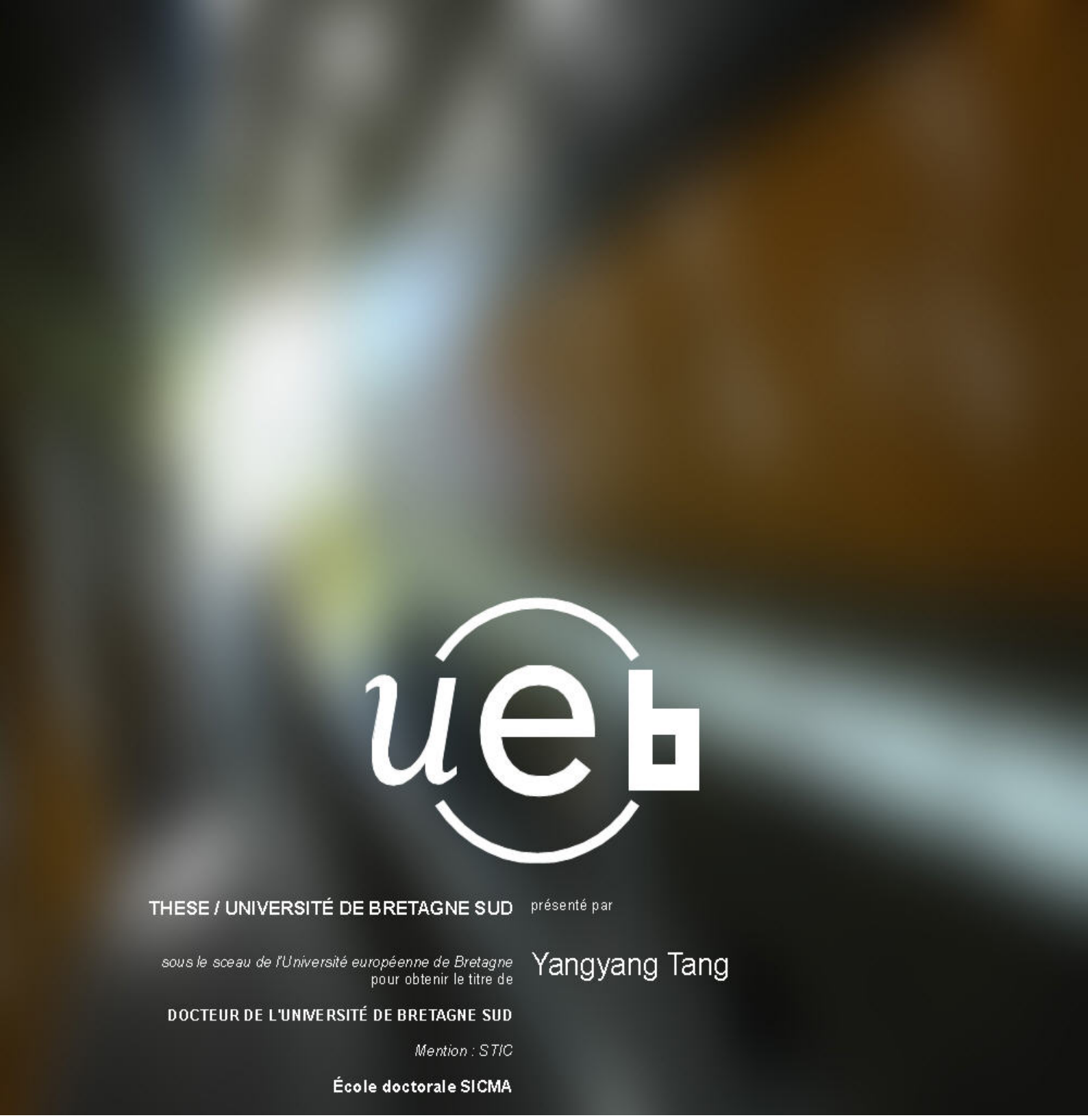


THESE / UNIVERSITÉ DE BRETAGNE SUD

sous le sceau de l'Université européenne de Bretagne
pour obtenir le titre de

DOCTEUR DE L'UNIVERSITÉ DE BRETAGNE SUD

Mention : STIC

École doctorale SICMA

présenté par

Yangyang Tang

Computation on Unreliable Architecture

Thèse soutenue le le 29 Janvier 2013,
devant la commission d'examen composée de :

Mme. Marie-Laure Boucheret
Professeur des Universités, Université de Toulouse / Rapporteur

M. Jean-Luc Danger
Professeur des Universités, Ecole Nat. Sup. de Télécommunications / Rapporteur

M. Jean-François Hélard
Professeur des Universités, Institut Nat. des Sciences Appliquées de Rennes / Examinateur

M. Chris Winstead
Associate Professor, Utah State University, Logan, UT / Examinateur

M. Emmanuel Boutillon
Professeur des Universités, Université de Bretagne-Sud / Directeur de thèse

M. Christophe Jégo
Professeur des Universités, Ecole Nat. Sup. d'E.I.T.M.M. de Bordeaux / Co-directeur de thèse

M. Michel Jézéquel
Professeur des Universités, Télécom Bretagne de Brest / Co-directeur de thèse

Résumé

En théorie, les circuits électroniques conçus selon la méthode du pire-cas sont supposés garantir un fonctionnement sans erreur pour un coût d'implémentation élevé. Dans la pratique les circuits restent sujets aux erreurs transitoires du fait de leur sensibilité aux aléas tels que la radiation et la température. En revanche, une conception prenant en compte la tolérance aux fautes permet de faire face à de tels aléas comme la variabilité du processus de fabrication. De plus, les erreurs transitoires et la variabilité de fabrication s'intensifient avec l'émergence de nouveaux processus de fabrication et des circuits de dimension de plus en plus réduite. La demande d'une conception intégrant la tolérance aux fautes devient désormais primordiale.

La présente thèse a pour objectif de cerner la problématique de la conception de circuits sur des puces peu fiables et apporte des contributions suivant quatre aspects. Dans un premier temps, nous proposons une méthode de tolérance aux fautes, basée sur la correction d'erreurs et la redondance à faible coût. Puis, nous présentons un critère bidimensionnel judicieux permettant d'évaluer la fiabilité et l'efficacité matérielle de circuits. Nous proposons ensuite un modèle universel qui apporte une tolérance avec fautes à redondance faible pour les systèmes logiques d'aujourd'hui et les architectures nanoélectroniques de demain. Enfin, nous découvrons un décodeur tolérant aux fautes transitoires internes.

Abstract

Although materials could be fabricated as error-free theoretically with a huge cost for worst-case design methodologies, the circuit is still susceptible to transient faults by the effects of radiation, temperature sensitivity, and etc. On the contrary, an error-resilient design enables the manufacturing process to be relieved from the variability issue so as to save material cost. Since variability and transient upsets are worsening as emerging fabrication process and size shrink are tending intense, the requirement of robust design is imminent.

This thesis addresses the issue of designing on unreliable circuit. The main contributions are fourfold. Firstly a fast error-correction and low cost redundancy fault-tolerant method is presented. Moreover, we introduce judicious two-dimensional criteria to estimate the reliability and the hardware efficiency of a circuit. A general-purpose model offers low-redundancy error-resilience for contemporary logic systems as well as future nanoeletronic architectures. At last, a decoder against internal transient faults is designed in this work.

n d'ordre : XX

Université de Bretagne Sud

Lab-STICC

Centre de Recherche Christiaan Huygens - BP 92116, 56321 Lorient CEDEX

Tél : + 33(0)2 97 87 45 62 Fax : + 33(0)2 97 87 45 27

Remerciements

Je remercie Mme. Marie-Laure Boucheret, Professeur des Universités, Université de Toulouse et M. Jean-Luc Danger, Professeur des Universités, Ecole Nat. Sup. de Télécommunications, d'avoir bien voulu accepter la charge de rapporteur.

Je remercie M. Jean-François Hélard, Professeur des Universités, Institut Nat. des Sciences Appliquées de Rennes et M. Chris Winstead, Associate Professor, Utah State University, Logan, UT, d'avoir bien voulu accepter la charge d'examinateur.

Mes plus sincères remerciements à M. Emmanuel Boutillon, Professeur des Universités, Université de Bretagne-Sud, M. Christophe Jégo, Professeur des Universités, Ecole Nat. Sup. d'E.I.T.M.M. de Bordeaux et M. Michel Jézéquel, Professeur des Universités, Télécom Bretagne de Brest, qui ont dirigé ma thèse. Leur aide, conseil et guide, m'ont été précieux pendant ces trois années.

Je remercie Chris Winstead, Associate Professor de Utah State Univeristy, qui m'a permis d'apprendre beaucoup sur la science et aussi d'avoir une expérience inoubliable.

Je remercie sincèrement enfin mes parents, qui me donnent tourjours de l'espoir afin de me pousser plus loin. Je remercie aussi Xiaowei et Patrick qui sont tourjours la pour moi, Eva, Li Jing, Zhang Yang, Gopalakrishnan Sundararajan, et aussi Maria pour son amitié.

Contents

General Introduction

Due to the rapid development of logic circuit manufacturing in the last two decades, the electronic device has been entitled deep nanoscale. As digital technology approach the limits of planar integration, device-level reliability has emerged as a critical concern. Above all, the fluctuation and variability issues induced during the fabrication further threaten the design reliability via functional failures. Densely integrated CMOS circuits are increasingly affected by thermal upsets which induce momentary, transient logic faults in digital architectures. The problem of transient upsets is compounded with the three-dimensional integrated circuits, and with the emergence of "post-CMOS" nanoscale devices that perform computations using a very small number of electrons.

Faults experienced by semiconductor devices fall into three main categories: permanent, intermittent, and transient. Permanent faults are caused by manufacturing defects or device wear-out [SH04]. They reflect irreversible physical changes. Intermittent faults occur because of unstable or marginal hardware. They can be activated by environmental changes, like higher or lower temperature and voltage variation [ENW08]. Most of the time, intermittent faults precede the occurrence of permanent faults. The transient defects and faults are becoming major impact on the reliability of electronic technologies. Due to the size downscaling processes, digital logic circuits are increasingly vulnerable to cross-talk, high-energy particle collisions and radiations [Bau01], thermal noise, electromagnetic interference (e.g. from switching events in neighboring devices on the same chip), timing failures in synchronous pipelines, power supply noise, and other sources [KP74, ENW08, SH04]. Among these transient faults, radiation-induced Soft-Error (SE) has led people to high attention on the topic of cosmic radiation induced transient faults in nanoscale computation. This event is called Single-Event Transient (SET) or Single-Event Upset (SEU) [Bau01] (See further information for the SET issue in Annex. A.1). For all these reasons, it has been proposed to embed error-correcting logic within integrated digital systems [SAKJ10, LCB+10].

0.1 Related Works

In the literature, a considerable amount of approaches has been proposed to mask or mitigate SE in different contexts. The well-known Triple Modular Redundancy (TMR) technique and similar conventional techniques that duplicate module in spacial or temporary style, as well as parity coding, require majority voting afterwards [Lal85]. In Error Correcting Code (ECC) context. More informations about the ECC technique can be found in Annex. A.2. It is well known that ECC has been used in non-volatile memory [RMR08], as well as in SRAM [SYH+06] and DRAM [BAS05]. In a software-only context, the recovery technique is carried out by instruction-level lockstep or multithreading instead of hardware redundancy [CRA06, RM00]. In Redundant Residue Number System (RRNS) context, a Bi-directional RRNS (BRRNS) that requires redundant moduli satisfy some constraints to achieve fast error-correction with minimum

hardware overhead, as one of our previous work [TBJJ10]. Moreover, two other contexts can be held up: the matrix multiplication approach in algorithm level [LCAP08], the add/remove functionally redundant wire approach [AM08] towards synthesis context.

At architectural level, Shim *etal.* [SSS04] presented Algorithmic Noise-Tolerance (ANT) technique to achieve reliable low-power digital signal processing with a reduced precision replica. An ANT-based system, Algorithmic Soft-Error Tolerance (ASET) [SS06] is composed of an error-control block that detects and corrects errors in the main function block has been developed. Since the arithmetic units employed in Digital Signal Processing (DSP) systems (in [SSS04]) are based on Least Significant Bit (LSB) first computation, soft errors appear first in the Most Significant Bit (MSB), resulting in errors in a large magnitude. In this way, the reduced precision block can be achieved with reduced precision operands. However, ANT method is hardly adapted as a general-purpose model while the reduced precision block can not be done in an efficient way. Likewise, ASET is constrained by the construction of an estimator which provides the redundancy required for error detection and correction. An Error Resilient System Architecture (ERSA) [LCB+10] has been designed for low-cost robust system, but with a higher cost in terms of hardware complexity for general-purpose applications. The concept of ERSA is assigning some tasks for two types of cores: bunch of undependable cores as slave functions for fast energy-efficient but error-vulnerable calculations and other dependable cores as master functions.

Note that in some cases, the unit for detecting or correcting errors is usually regarded as error-free. Embedded error correction may be used to mask permanent circuit defects as well as transient faults. Unfortunately an embedded error correcting decoder has to be designed using the same error-prone devices as the functions to correct. This introduces the problem of reliable decoder design that is able to detect and/or correct faults in its own operations. There is a long list of works that addresses this problem, based on solutions in the form of "self-checking checkers." Self-checking logic circuits are traditionally designed using formal logic methods, where it can be proved that the circuit detects or corrects up to a fixed number of faults.

Traditional masking mechanisms [KH04] can be found in the category of circuit level solutions. Such like, three masking techniques in circuit-level. Temporal masking [LDJK94], that is based on a latching window for latch components, in which case the error will not be latched while the transient propagates towards a sequential element; Logical masking [CP93], since the input of a logic gate is not sensitized to an error event, it completely masks the erroneous propagation. For instance, If SET happens on an input to a NAND (NOR) logic gate, but one of the other inputs is in the controlling state, thus the SET is masked; Electrical masking [DL95], while the CMOS circuits limit bandwidth, transients with the bandwidth higher than a threshold frequency can be weaken. Thus, SET pulse will be vanished through a sequential path.

A sophisticate flip-flop, referred as to Razor flip-flop, detects the signal delay errors and tunes supply voltage for power aware computing purpose. [EKD+03]. A Built-

In Soft Error Resilience (BISER) technique was proposed for the corrections of SEs in latches, flip-flops and combinational logic [ZMM+06b, ZMM+06a, MZS+07]. BISER is considered as a Dual Modular Redundancy (DMR) application by using Muller C-elements as the decision maker. Muller C-element was proposed by Muller in 1959 [MB59], that is increasingly used for fault-masking in digital circuits [GR03, JM05b, JM05a, WGRS05, WH09, Win09, WEH09, WLMT11]. It attracts us more attention for robust system design as well [TWB+12, WTB+12].

Although materials could be fabricated as error-free theoretically with a huge cost for worst-case design methodologies [fS], the circuit is still susceptible to transient faults. Much efforts have been made by designers for recovering performance after manufacturing. Since any embedded fault-masking solution is implemented using the same devices as the logic it protects, it is important for a fault-masking mechanism to be tolerant of internal errors within its own logic.

0.2 Contributions and Outline

Classically, all the processes in electronic digital circuits can be classified into three main categories: storage units, such as the memories; communication units, such as the buses; and computing units, such as the logic gates or adder. The scheme of an operation process without applying ECC strategies can be described as the non-shaded parts of Fig. 1. The result g is obtained from an operation $F(x)$ by given the data x as input.

In this thesis work, we put our emphasis on the application of ECC on unreliable circuit. A formal model for the application of ECC on unreliable circuit is illustrated as the shaded parts in Fig. 1. The model consists of three blocks: the encoding function $E(x)$ to produce a codeword $c \in \hat{C}$, the computing function $F_c(c)$ to perform the operation and the decoding function to recover the result g from an error-correction process applied to the received data y. Please note that all the three blocks are vulnerable to transient faults and permanent defects. As a result, in order for the ECC block to recover the result from the output of computing function $F_c(c)$, the output of function $F_c(c)$, y, has to be a linear code. To achieve this end, there are four cases. We introduce the four cases as the contributions of this thesis work presented as follow.

0.2.1 The Contributions by Order of Outline

- **Chapter 1**

  To obtain a linear code y, the first case is that a group homomorphism is obtained among $E(x)$ and $F_c(c)$. More precisely, the encoding function E is considered as a morphism of $\hat{S} \rightarrow \hat{C}$ and the computation F_c is the operation in $\hat{S}$. Fig. 2 shows the diagram of the application of ECC on unreliable computation for this case.

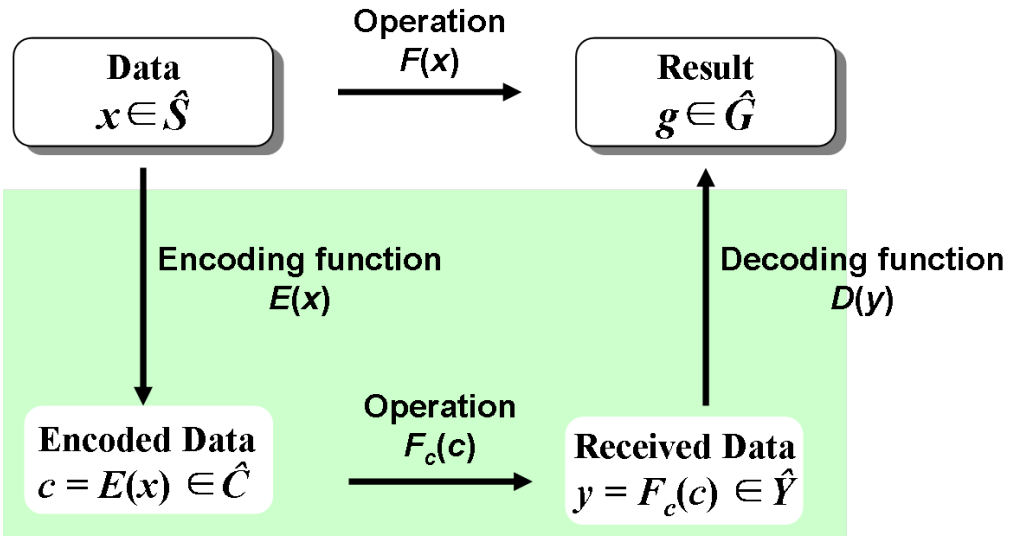


Figure 1: A formal model for the application of ECC on unreliable circuit. The non-shaded parts are the scheme of an operation process without applying the ECC strategy.

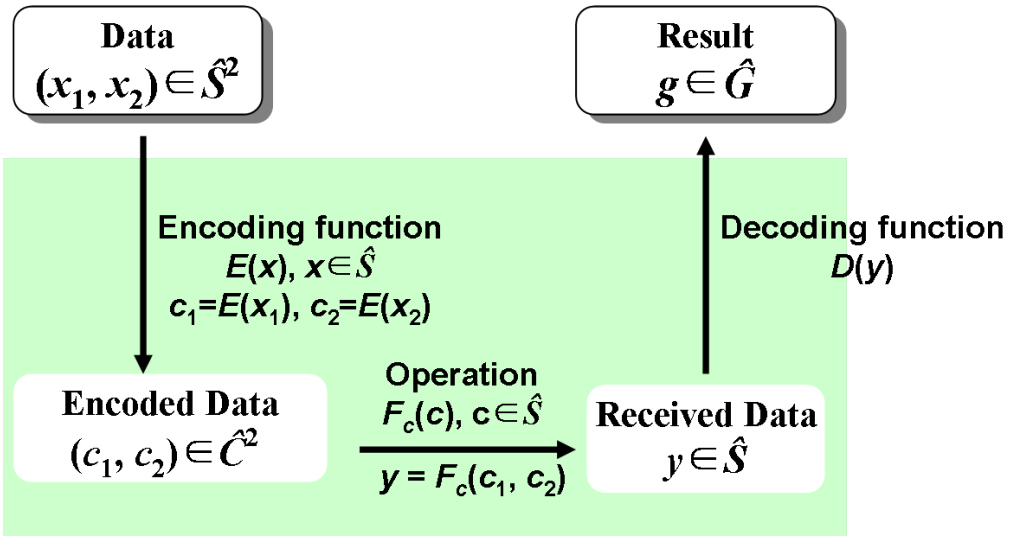


Figure 2: A formal model for the application of ECC on unreliable computation when the encoding function E is considered as a morphism of $\hat{S} \rightarrow \hat{C}$ and the computation F_c is the operation in $\hat{S}$.

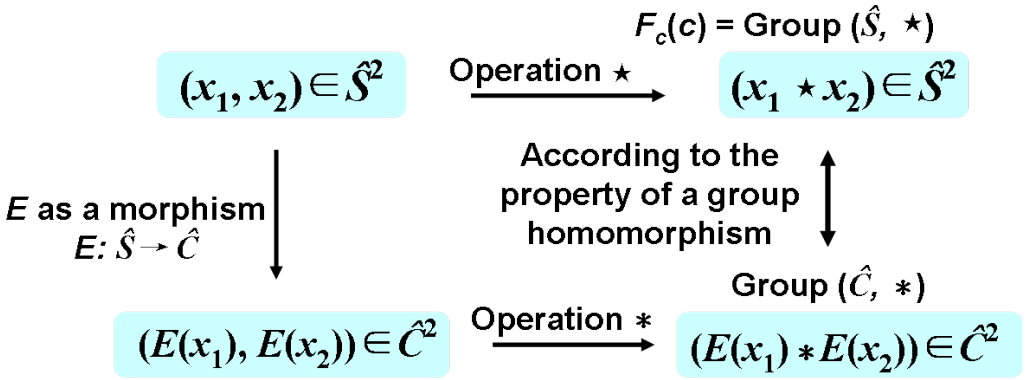


Figure 3: Diagram of the property of a group homomorphism, where E is considered as a morphism of $\hat{S} \rightarrow \hat{C}$ and the computation F_c is the operation in $\hat{S}$.

Definition 1. *Given two groups ($\hat{S}$,$\star$) and ($\hat{C}$,$*$) ($\star$ and $*$ represent their operations), we define that a group homomorphism from ($\hat{S}$,$\star$) to ($\hat{C}$,$*$) is a morphism E: $\hat{S} \rightarrow \hat{C}$ such that for all x_1 and x_2 in $\hat{S}$, it holds that*

$$E(x_1 \star x_2) = E(x_1) * E(x_2) \tag{1}$$

Hence, the operation F_c in $\hat{S}$, $(x_1 \star x_2) \in \hat{S}$, is equivalent to the operation in $\hat{C}$, $(E(x_1) * E(x_2)) \in \hat{C}$, expressed as

$$(x_1 \star x_2) \in \hat{S} = (E(x_1) * E(x_2)) \in \hat{C} \tag{2}$$

Moreover, Fig. 3 shows the diagram of this property of a group homomorphism.

For example, the redundant residue number system satisfies the constraint of a group homomorphism. Its encoding process is carried out by the modulo operation and its computing function is done by the operation of residues. In this case, both functions are linear and thus (1) and (2) are satisfied.

For a (6,3) RRNS code, the computation $F_c(c)$ is an addition of two digits, x_1 and x_2, $x_1 = x_2 = 20, \in \hat{S} = [0, 692]$, and the moduli set is (7,9,11,13,16,17). Firstly the encoding function $E(x)$ performs the modulo operations for both x_1 and x_2. Thus, the codewords c_1 and c_2 $(c_1, c_2) \in \hat{C} = [0, 16]$ are obtained as the residue vector (6,2,9,7,4,3) for both x_1 and x_2. The addition for x_1 and x_2 is performed by the addition of c_1 and c_2 to produce the received data $y \in \hat{S} = [0, 692]$ after the computation. In this way, y is obtained as a vector of residues (5,4,7,1,8,6). Finally, the sum of the two digits, x_1 and x_2, can be restored from y, thanks to an iterative decoding process.

Let us commute the operation of the composite function. Instead of the modulo operation (as the encoding) before the addition (as the computation), the addition

is done at first. Thus, the output of the addition ($x_1 + x_2$), 40, is obtained. After the modulo operation for 40 with the same moduli set (7,9,11,13,16,17), the received data y is thus calculated as (5,4,7,1,8,6).

Consequently, the redundant residue number system is an example of the case where both encoding function $E(x)$ and computing function $F_c(c)$ are linear. Therefore, (1) and (2) are satisfied since a group homomorphism is obtained. Please note that that the sets of the inputs and the outputs for both function F_c and function E should be compatible.

To investigate this approach, chapter 1 introduces a fast arithmetical fault-tolerant method based on residue number system (RNS) [Gar59] for more robust computations. The RNS provides a very fast arithmetic due to its capability of performing the carry-free operations, i.e. addition, subtraction and multiplication. By adding some redundant residues, the RNS has an error detection and correction property that is called Redundant Residue Number System (RRNS) [ST67]. However, the data restoration of a conventional RRNS is an iterative decoding process. In this case, it requires high latency to achieve an error-correction. In this chapter, we propose a technique, referred as to bidirectional redundant residue number system (BRRNS) [TBJJ10], that achieves fast arithmetic and enables to mask an overall SET. In this method, the detection and the error-location are simultaneously performed in a plural parallel consistent-checking that has the capability of locating the corrupt digit. Therefore, a fast error-correction is obtained. An efficient pipeline architecture for the BRRNS based ripple-carry adder is illustrated as well.

- **Chapter 2**

  Secondly, when the received data y is obtained from the repetitions of computation $F(x)$, it is a linear code as well, as shown in Fig. 4

  For the Triple Modular Redundancy (TMR) method [LV62], its encoding function is obtained by two duplications of the original computing function. In this case, the received data of the decoding function are the outputs of the original function and of the two duplications as well. Obviously, the received data is linear. Moreover, the well-known repetition code [Mac03] is also an example of this technique. Actually, the encoder of the repetition code repeats, n times, a particular information bit. As such, if we have a (3,1) repetition code with $n = 2$, then, it is equivalent to a triple modular redundancy method.

  TMR has been employed as the fault-tolerant technique for a robust system since fifty years ago. However, is it efficient to apply a triple modular redundancy technique to increase the robustness of an unreliable computation? Additionally, how can we make a fair comparison between two fault-tolerant methods?

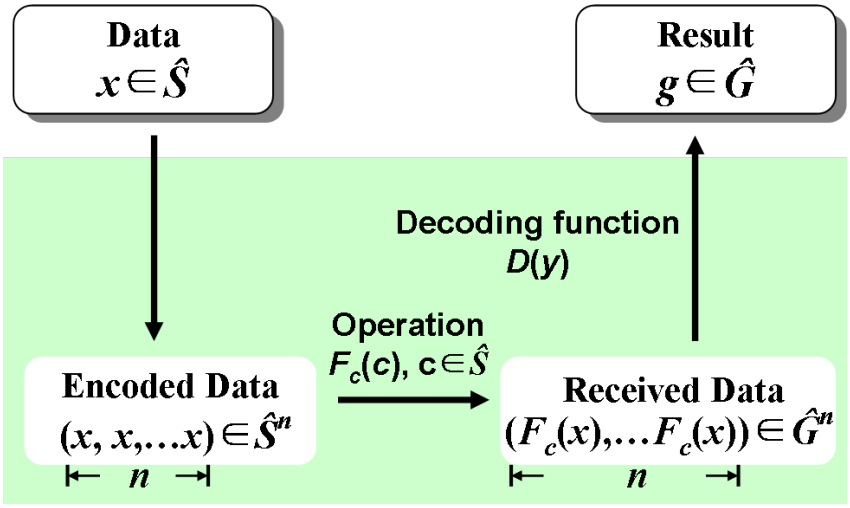


Figure 4: A formal model for the application of ECC on unreliable computation when received data y is obtained from the repetitions of computation $F(x)$.

In chapter 2, we present a novel criterion, called Reliability-Efficiency Criteria (RE-Criteria). It defines a two dimensional space of solution, that consists in: error probability at fanout and hardware efficiency estimations. In this work, by taking into account that the correction unit is also subject to error, we propose to assess the quality of an architecture by using not only its efficiency (i.e. the normalized number of operation per area and per time unit), but also its final output bit error rate. By exploiting this composed dual space criteria, designers are able to discern the properties, the reliability and the efficiency, of different fault-tolerant strategies.

- **Chapter 3**

  The third case is that the processes in the circuit are storage or communication. Thus, the ECC strategy can be directly applied, as shown in Fig. 5.

  A Fast Decoding Method for Binary Symmetric Channel

  A solution to the subject of embedded binary error-correction is presented in Chapter 3. A modified version of Gallager's well-known Bit-Flipping decoding method has been proposed for decoding LDPC codes over a Binary Symmetric Channel (BSC) in [CV09]. A presented modification is about the estimate of a variable node that is taken to be the majority of the incoming messages at the end of each iteration. In this work, we propose a similar modification, to define the the majority of the incoming messages only during the final decoding iteration, instead of at the end of each iteration. Moreover, we prove that the modified

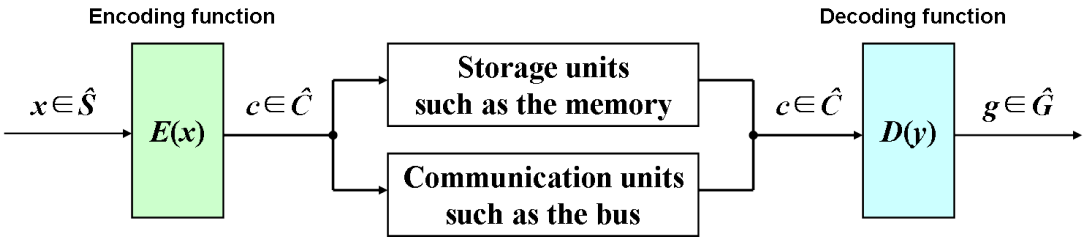


Figure 5: A formal model for the application of ECC on unreliable computation when the operations are storage units or communication units.

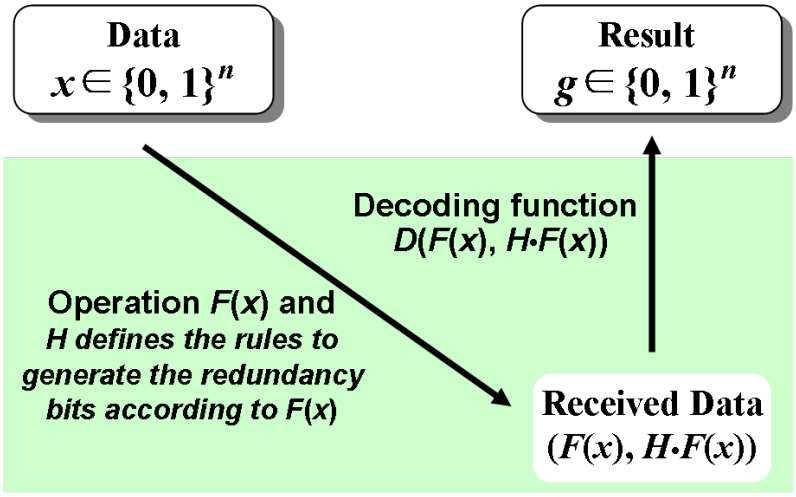


Figure 6: A formal model for the application of ECC on unreliable computation when the redundancy is directly generated from a parity-mapped function.

version can increase the decoding process via density evolution analysis. Since slight extra hardware resource overhead is required, our method is appropriate for fast low-power error-correction design.

The fourth case is to directly generate the data y for the decoding function, by a concatenation of two outputs from the original function and a parity-mapped function. In this case, y is assured as a linear code, as shown as the diagram in Fig. 6.

This model, called coded Dual-Modular Redundancy (cDMR), is based on the Dual-Modular Redundancy (DMR). Note that a parity-mapped function replaces a duplicated original function. The cDMR is a modified version of an error-correction model, referred as to Low-Density Parity-Check (LDPC)-coded Fault Compensation Technique (LFCT), that was proposed by Winstead in [WH09].

A General Embedded Error-Correction Model, called cDMR

The LFCT method is applied to reliably compute some function $F(x)$. Due to the affections by transient/permanent faults, the output of $F(x)$, s, is not reliable. Since an LDPC code is defined by a parity-check matrix H, the valid codeword as a vector $[s\ r]$ can be generated through another encoding function. The sequence r is the redundancy bits of s. Subsequently, a stochastic decoder [GR03,WGRS05] has been proposed to correct some errors that occur at the output of some logic computations s. Although a considerable gain in terms of decoding performance at the output is rendered by LFCT, a critical drawback would lead the method to fail. If the correlated errors occur at the output of computing function $F(x)$ and as well encoding function, the stochastic decoding can not perform an efficient error-correction.

In this chapter, to take into account the occurrence of correlated errors, we present a modified version of LFCT, cDMR. Thanks to the cross-bar technique, in cDMR the encoding function to generate the redundancy bits r is realized in a correlated-error-free fashion. Moreover, the cDMR is modularized in a formal general robust system.

- **Chapter 4**

  This embedded error-correction model, cDMR, can be adapted into any unreliable design whenever two issues are avoided. The first is that correlated errors in original function's output and also the parity-mapped function's output. The second is that the ECC circuit is able to perform efficient error-correction in the presence of internal transient faults. Thanks to the cross-bar technique, the correlated-error issue can be avoided in cDMR. In this chapter, we focus on an efficient error-correction in the presence of internal transient faults within decoder itself.

  A Decoder Against Internal Transient Faults

  Muller C-element based Decoder (MCD) is a decoding technique that is able to perform error-correction in the presence of hight rate of internal transient faults. The MCD is an improved version of GBF, based on the Muller C-elements to carry out the variable node process. In practice, for the decoding process of MCD, the messages that are sent from variable nodes to check nodes are more reliable thanks to the processing of variable node by C-elements. By comparison to the GBF, when the internal error rate is as high as 0.01, the MCD does not exhibit a Bit Error Rate (BER) performance degradation thanks to an increase of the number of decoding iterations. Therefore, the MCD can be considered as a decoder who inherits the potential of error-resilience to transient upsets of Muller C-elements.

  A Space-Time Redundancy Technique to Improve the Decoding Performance

  To further increase the BER performance of a decoder in the presence of internal faults, the Space-Time redundancy technique can be applied. In fact, a major-

ity voter at the end of decoding step enables to sample each output during the last partial decoding iterations. Then, it outputs the decision from a majority of sampled data. Thanks to the Space-Time technique, the transients that induced flipped over at the decoded output is able to be masked by the correct outputs. In addition, the proposed technique can be widely adapted for any message-passing decoder to further improve the BER performance in the presence of internal errors.

Moreover, by examining the error-resilience from state memory of C-element, we study the capability of fault-tolerance inherited by C-element. In addition, we further investigate the behaviors of the decoder against internal faults, referred as to MCD, via density evolution theory [RU01] and trellis structure [Ung82].

Finally, we introduce a good decoder candidate for ECC of cDMR: a MCD of (3,6) short-length LDPC code. Thus, when the decoding process is error-free, the MCD achieves also a notable gain in terms of decoding performance by comparison with the GBF. As such, it introduces an error-correction benefit in a high error-rate of internal faults.

1

Fast Fault-Tolerant Arithmetic for Embedded Computation

In this chapter, a fast arithmetical fault-tolerant method is presented for robust computation (i.e. addition, subtraction, multiplication, and the applications residue number system based Fast Fourier Transform (FFT) and Filters, and etc.). Based on the residue number arithmetic, the proposed technique, called bidirectional redundant residue number system, is able to cope with single SET. It requires that the redundant moduli satisfies some constraints to achieve fast error correction.

In this system, the detection and the diagnosis are simultaneously performed in plural parallel consistent-checking that has the capability of locating the corrupt digit. For this sake, fast error-correction is applied. Moreover, by restoring the data via two independent information sources, a single SET can be masked under single error-correction operation in the presence of internal transient faults. Furthermore, an efficient pipeline architecture for the self-diagnosis decoder is proposed as well.

1.1 Introduction

Since any embedded fault-masking solution is implemented using the same device as the logic it protects, it is important for a fault-masking technique to be tolerant of internal errors within its own logic. Thus, the restoring block is supposed to perform error-correction under a noisy process. Researchers previously investigated the performance of a variety of error-correcting schemes under the influence of intrinsic faults [SSS04, SS06, LCB+10, WLMT11, WH09, TBJJ10]. The concept of bidirectional arithmetical fault-tolerant method presented in this chapter is using two independent information sources and restoring the data via two information sources with a sophisticated mechanism, achieving a fast error-correction.

This concept is similar to Algorithmic Noise-Tolerance (ANT) technique presented by Shim *etal.* in [SSS04]. In general, to increase the reliability of a computation function, an additional reduced precision replica of the original computation function is employed. This replica is able to give an reduced precision estimation for the output of computing function. By applying some criteria to make the decision, the final out-

put is chosen between the output of the computation function and the output of the replica. Since the reduced precision replica consumes much less power than the original function, the ANT achieves reliable low-power digital signal processing. However, the perturbation of the replica can cause the failure of the system.

An ANT-based system, Algorithmic Soft-Error Tolerance (ASET) [SS06] is composed of an error-control block that detects and corrects errors in the computation function block has been developed. By comparison to the original ANT technique, two replica modules of the computing function are duplicated to produce two estimations to the output of the computing function. Thus the error-control block makes the decision among the output from original function and the two estimations via a sophisticated metric. Since the arithmetic units employed in Digital Signal Processing (DSP) systems (in [SSS04]) are based on Least Significant Bit (LSB) first computation, soft errors appear first in the Most Significant Bit (MSB), resulting in errors in a large magnitude. In this way, the reduced precision block can be achieved with reduced precision operands. However, ANT method is hardly adapted while the reduced precision block is difficult to obtain. Likewise, ASET is constrained by the construction of estimator which provides the redundancy required in error detection and correction.

In addition, an Error Resilient System Architecture (ERSA) has been addressed for low-cost robust system in [LCB+10]. The ERSA assigns tasks for two types of cores: bunch of undependable cores as slave functions for fast, energy-efficient but error-vulnerable calculations; and certain dependable cores as master functions. However, it takes a higher costs for general-purpose applications.

The rest of this chapter is organized as follows: we first review some well-known FT techniques to this subject (a traditional approach, triple modular redundancy, ANT based techniques, and ERSA model). Then, the residue number system and redundancy residue number system are detailed, respectively. The proposed fast recovering method for the reliable arithmetic computation and its pipeline architecture are presented afterwards.

1.1.1 Triple Modular Redundancy

Triple Modular Redundancy (TMR) [LV62] is a majority-logic technique developed in the 60's which has been widely used for electronic systems in the space and nuclear industries to ensure reliable operations. Although TMR has fallen out of the method to cope with dependable issue, it is still employed today for applications that require high reliability. In the standard TMR scheme, a logic operation is duplicated thrice, usually by three parallel independent modules in the case of spatial redundancy or by reusing a single modules in the case of temporal redundancy, as shown in Fig.1.1. Then, for each logic output, there are three values, each of which may be correct or incorrect. The probability of error on each value P is assumed to be independent from that of the other values, and is equal to a small constant value. The TMR output is obtained by taking the majority value. The output is thus correct if any single error occurs

among those three, but two or more simultaneous errors will cause an error. Hence, TMR reduces the resulting error probability to $3P^2 - 2P^3$. It induces twice hardware redundancies of a single modular [TBJJ11]. Its reliability and hardware efficiency is further analyzed in Section.2.

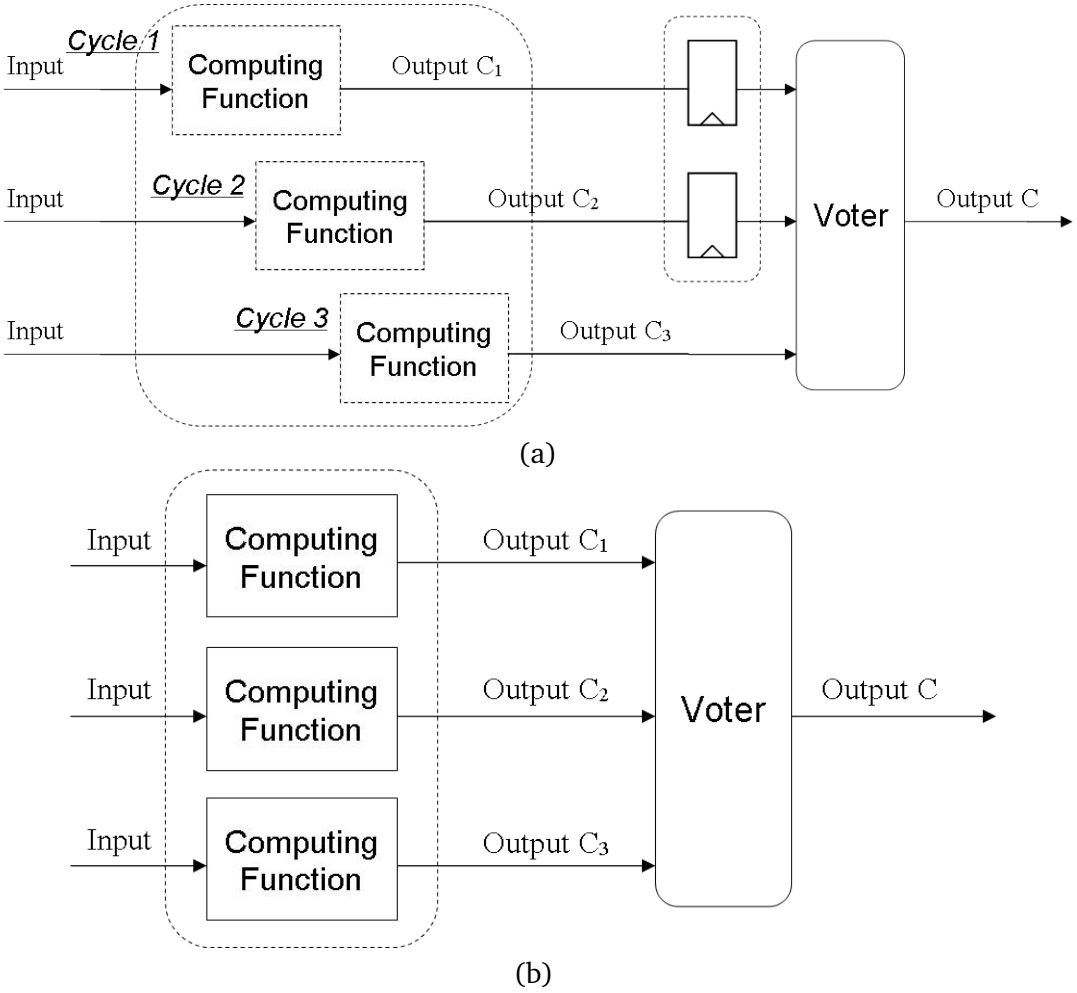


Figure 1.1: Two solutions for TMR: (a) Temporal TMR, (b) Spatial TMR.

1.1.2 Algorithmic Noise-Tolerance

Shim *et al.* [SSS04] presented Algorithmic Noise-Tolerance (ANT) technique to achieve reliable low-power digital signal processing with a Reduced Precision Redundancy (RPR). Fig. 1.2-(a) demonstrates an ANT-based system scheme, where a RPR based replica is employed to generate redundancy to be fed into error-correction block, as the shaded area in Fig. 1.2-(a). More precisely, in Fig. 1.2-(a), the main block represents any function under Voltage Overscaling (VOS) technique which provides an energy sav-

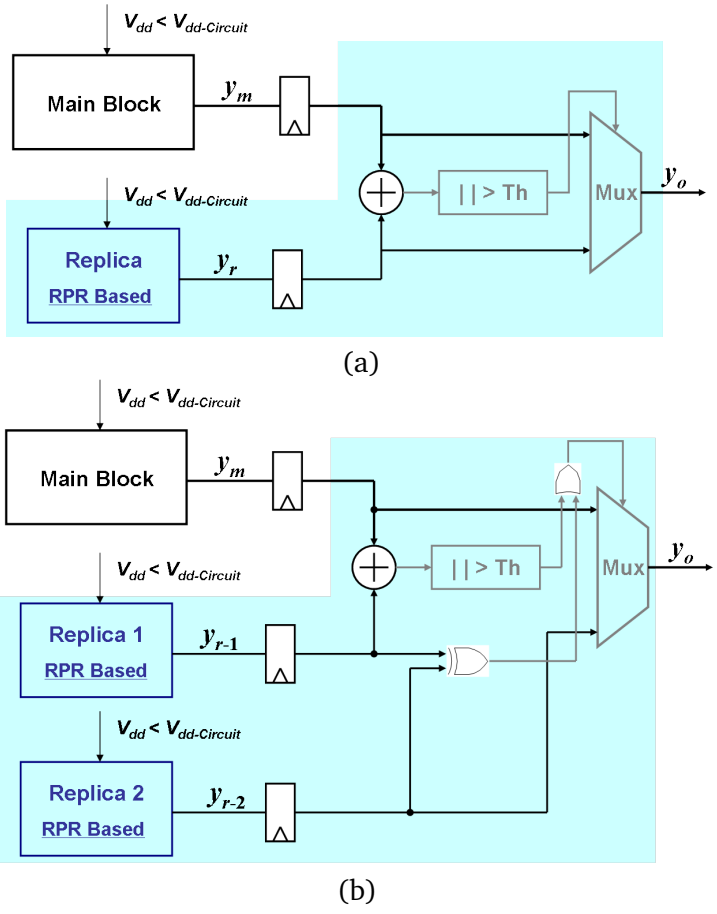


Figure 1.2: Two reduced precision redundancy based fault-tolerant technique schemes: (a), single estimator (b), double estimator.

ing strategy. Due to the application of VOS and the occurrence of transients, the main block output y_m may be erroneous. Such that, a low-complexity that approximately computes an output y_r that is expected as the output of error-free main block. The shaded area, error-correction block, calculates the difference d_e between y_m and y_r, thanks to an Euclidean metric $d_e = |y_m - y_r|$. To restore the data in the multiplexer, the metric of decision y_o is described as follows:

$$y_o = \begin{cases} y_m, & \text{if } d_e \leq Th \\ y_r, & \text{otherwise} \end{cases} \tag{1.1}$$

where the Th is a pre-specified threshold. However, the occurrence of transients in the estimator leads the scheme in Fig. 1.2-(a) undependable.

To stem the disturbance by noisy estimator, an ANT based approach, called Algorithmic Soft-Error Tolerance (ASET) has been proposed in [SS06]. In fact, instead of using a single RPR estimator, ASET employs two RPR estimators to produce more

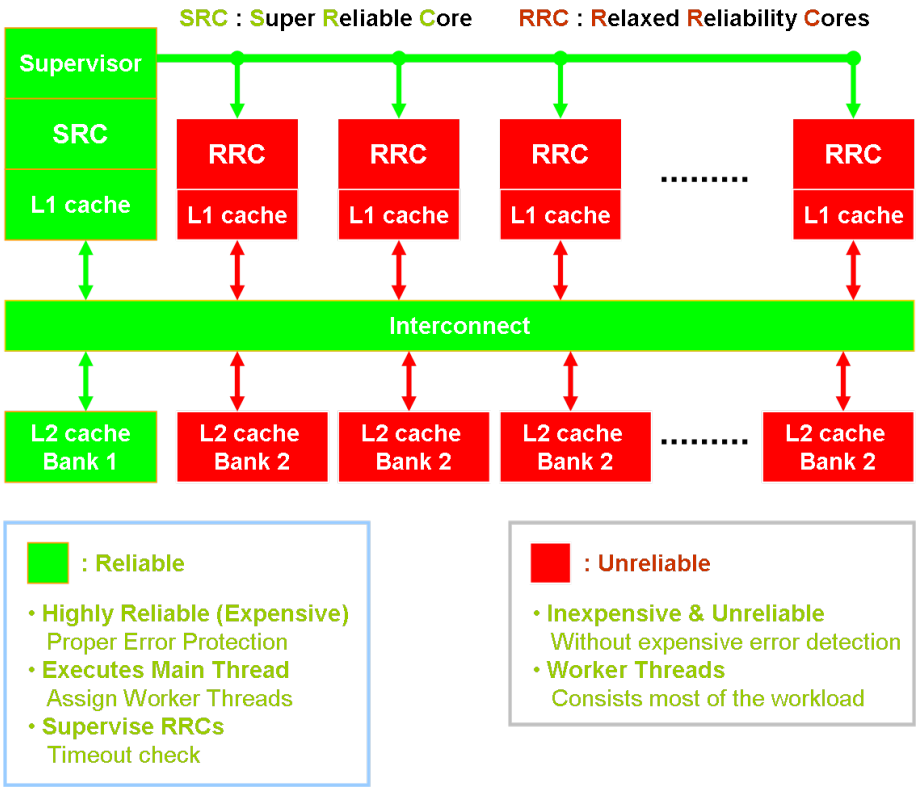


Figure 1.3: An example of the Error-Resilient System Architecture (ERSA). The system is realized by two types of components, reliable blocks and unreliable blocks. The reliable ones dominate the main threads that require high precision. The unreliable ones occupy the worker threads that require low precision.

reliable redundancy. It enables to take a more reliable decision after. Fig. 1.2-(b) shows a main block that is paralleled with two RPR estimators. To take into account the transients that occur in estimators, two differences are calculated: $d_e(y_m, y_{r-1})$ and $d_h(y_{r-1}, y_{r-2})$. $d_e(y_m, y_{r-1})$ is the Euclidean metric between y_m and y_{r-1} and $d_h(y_{r-1}, y_{r-2})$ is the Hamming distance between y_{r-1} and y_{r-2}. The error control metric is thus summarized as follows:

$$y_o = \begin{cases} y_m, & \text{if } d_e(y_m, y_{r-1}) \leq Th \\ y_m, & \text{if } d_e(y_m, y_{r-1}) > Th \text{ and } d_h(y_{r-1}, y_{r-2}) = 1 \\ y_{r-2}, & \text{if } d_e(y_m, y_{r-1}) > Th \text{ and } d_h(y_{r-1}, y_{r-2}) = 0 \end{cases} \tag{1.2}$$

However, two drawbacks limit the based ANT method to be widely applied. It is hardly adapted while the reduced precision based estimator can not be realized in a feasible way. Moreover, it requires that the decision block is reliable so as to perform a correct error-control.

1.1.3 Error-Resilient System Architecture

An Error Resilient System Architecture (ERSA) [LCB+10] has been addressed for low-cost robust system. As shown in Fig.1.3, two kind units comprise the system, reliable block and unreliable blocks as main thread and worker threads, respectively. In this case, while the main block is the only reliable unit, different robustness levels are configured among the workers' architectures. As a result, the ERSA solution is able to maintain high error resilience to high-order bit errors and control errors to low-order bit errors.

In short, the ERSA provides an asymmetric reliability based solution in many-core architectures system so that the system achieves high error resilience to high-order bit errors and control errors to low-order bit errors. Since high computing precision is not strictly requested, a low-cost robust system architecture is feasible. However, due to low reliability set up in some cores, the ERSA is not suitable to high accuracy system. In addition, a higher costs may be involved for general-purpose applications.

1.2 Redundancy Residue Number System

1.2.1 Introduction

The RNS, firstly introduced by Garner [Gar59], provides a very fast arithmetic due to its capability of performing the carry-free operations, i.e. addition, subtraction and multiplication. By adding some redundant residues, the RNS has an error detection and correction property that is called Redundant Residue Number System (RRNS). A method based on RRNS that is able to correct single bit error by encoding the residues was proposed by Cheney [Che62]. Szabo and Tanaka [ST67] dedicated their investigations to the properties of the RNS and presented a method for correcting single residue-error correction. The idea of consistency checking was first introduced by Watson and Hastings [WH66]. They also defined a method based on a table for single residue error correction.

1.2.2 Residue arithmetic fundamentals

Definition 2. *A standard RNS [Gar59] is characterized by a set of k pairwise relative prime positive integers, i.e. greatest common divisor* $(m_i, m_j) = 1$ *with* $i \neq j$, m_1, m_2, m_{k-1}, m_k, *called moduli. The order of moduli is in increasing, i.e.,* $m_1 < m_2 < \cdots < m_{k-1} < m_k$. *Their product represents the interval* $[0, M)$ *called the legitimate range that defines the useful computational range of the number system, that is,*

$$M = \prod_{i=1}^{k} m_i \tag{1.3}$$

If positive and negative numbers are considered, the dynamic range is defined as $[-(M-1)/2, (M-1)/2]$ if M is odd, and $[-M/2, (M/2)-1]$ if M is even. Every natural integer X in the legitimate range can be represented by a set of residues, r_1, $r_2 \cdots$, r_{k-1}, r_k, where

$$r_i \equiv |X|_{m_i} \; or \; r_i \equiv X(mod\ m_i) \tag{1.4}$$

with $i \in [1, k]$, and $|X|_{m_i}$ denotes X modulo m_i. Due to the carry-free property, the three operations, namely, addition, subtraction and multiplication, can be operated with respect to moduli independently, i.e.

$$x_1 x_2 \cdots x_k * y_1 y_2 \cdots y_k = z_1 z_2 \cdots z_k, z_i \equiv |x_i * y_i|_{m_i} \tag{1.5}$$

with $*$ denotes the three operations. Consequently, RNS approach is able to provide a fast arithmetic.

Definition 3. *An RRNS [Man72,BM73] is defined as an RNS with $n-k$ additional moduli. The moduli, m_1, m_2, m_{k-1}, m_k, are called the information moduli. The additional $n-k$ moduli, m_{k+1}, $m_{k+2} \cdots$, m_{n-1}, m_n, are called the redundant moduli. Moreover, the product of all the moduli, M_T, is denoted as the total range:*

$$M_T = \prod_{i=1}^{n} m_i = M M_R, M_R = \prod_{i=n-k}^{n} m_i \tag{1.6}$$

where M_R represents the illegitimate range. Similarly, the residue vector is composed of the information residues r_1, $r_2 \cdots$, r_{k-1}, r_k, and the redundant residues r_{k+1}, $r_{k+2} \cdots$, r_{n-1}, r_n. Any number belonging to the illegitimate range indicates the overflow or error occurrence as explained in the next subsection.

When the operations were performed in residue vector, a converter of residue to integer or binary (henceforth, termed simply residue/binary, vice versa, binary-residue) is necessary in order to build the number. It is well known that Chinese Remainder Theorem (CRT) and Mixed Radix Conversion (MRC) [ST67, Tay84, OP07] approaches are often applied for the conversion. Assume that an integer X formed in a residue vector r_i and $w_i = M_T/m_i$. According to the CRT, X then can be computed by

$$X = \sum_{i=1}^{n} w_i \left| c_i r_i \right|_{m_i} (mod\ M_T) \tag{1.7}$$

The coefficient c_i is determined by the congruence,

$$c_i w_i \equiv 1 (mod\ m_i) \tag{1.8}$$

It is often necessary to find the extra residue digits $r_{n+1} \cdots$, r_{n+p}, of an integer X for a spare set of relatively prime moduli $m_{n+1} \cdots$, m_{n+p}, given the residue vector r_1, $r_2 \cdots$, r_{n-1}, r_n, relative to a set of relatively prime moduli m_1, $m_2 \cdots$, m_{n-1}, m_n. This

is known as the Base Extension (BEX) [SK89] that is based on the CRT. Actually, X is firstly obtained by (1.8), and then the extra residue digits can be compute with (1.4).

Compared with the CRT, the MRC is carried out by a weighted approach. If the same previous example is considered, then the MRC is expressed by the following equations,

$$X = \sum_{i=1}^{n} a_i \prod_{j=1}^{i=1} m_j \tag{1.9}$$

where $0 \leq a_i < m_i$, and $\prod_{j=1}^{0} m_i = 1$.

The digits a_i are called the mixed radix digit. They can be obtained from a computational iteration, that is,

$$a_i \equiv \prod_{j=1}^{i-1} [m_{j,i}^{-1}(r_i - a_j)](mod\ m_i), \tag{1.10}$$

where $m_{j,i}^{-1}$ is defined as $m_{j,i}^{-1} m_j \equiv |1|_{m_i}$, with $a_1 = r_1$.

Thus, the MRC approach is less complex than the CRT approach thanks to a serial process. This process can be achieved in pipeline procedure for hardware design, which will be demonstrated in Section. 1.4.

1.2.3 The error detection and correction

1.2.3.1 RRNS Based Fault-Tolerant Model

Error-Correcting Codes (ECC) consist of an information part and a redundant part [McE02]. Having inherited the property of RNS, RRNS codes have the capability of allowing fast arithmetic. They have also the error detection and correction due to the extra redundant residue. Some works about the combination of RRNS and ECC has been done [YH01]. Assuming the SET occurs during the computation or in the memories (DRAM, SRAM, etc.), the fault-tolerant models applied the RRNS technique as shown in Fig.1.4. By applying the noisy channel coding theorem developed by Shannon [Sha48], the RRNS codes are composed by the binary/residue conversion as the encoder and the conversion as the decoder. Moreover, the computing function or the memory correspond to the noisy channel if they are affected by a SET [vN55, Var11]. For the sake of facility, we note that the residue/binary conversion is assumed as an error-free process hereinafter.

1.2.3.2 The error detection

The conventional approach for the error detection and the error correction is performed by the consistent checking or the BEX. The consistent checking was firstly proposed by Watson and Hastings in [WH66]. It can be defined as: all the residues in the residue

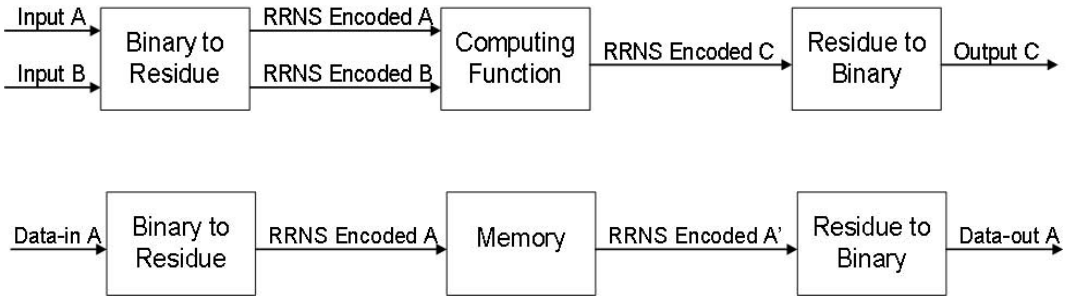


Figure 1.4: RRNS based fault-tolerant models: (a) for a computing function. (b) for a memory.

vector are correct if the condition

$$0 \leq X < M \tag{1.11}$$

is satisfied. The process that enables to determine whether or not it holds is called consistent checking.

Theorem 1. *The RRNS codes with $n - k = 1$ redundant modulus detect all single errors if the condition $m_1 < m_2 < \cdots < m_k < m_{k+1}$ is satisfied.*

Proof 1. *Suppose a single SET distorts the i^{th} residue digit in the residue vector of an integer X, it is represented as an addition of the number e_i to the residue while $0 < e_i < m_i$, namely,*

$$X' \longrightarrow [r_1, r2 \cdots, (r_i + e_i) \cdots, r_{n-1}, r_n] \tag{1.12}$$

if X' is the altered number. According to 1.5, the X' can be defined as the sum of the X and an integer E formed by the residue vector e_i,

$$X' = X + [0, 0 \cdots, e_i \cdots, 0, 0], [0, 0 \cdots, e_i \cdots, 0, 0] \equiv E(mod\ m_i) \tag{1.13}$$

With (1.7) and (1.9), E can be expressed by

$$E = c \prod_{j=1, j \neq i}^{n} m_j \tag{1.14}$$

where c is a non-zero multiplicative coefficient.

Since the redundant modulus m_{k+1} is larger than all the information moduli, E is larger than the legitimate range M in (1.3), namely,

$$\prod_{j=1,j\neq i}^{n} m_j \geq M = \prod_{j=1}^{n-1} m_j, n \geq i$$

And thus,

$$X' \geq X + cM \tag{1.15}$$

With the expression of the X' 1.15, the number E leads the decoded integer to appear in the illegitimate range. Hence, the single error is detected.

Theorem 2. *For MRC case, the RRNS codes with $n-k=1$ redundant modulus detect all single errors if the condition $a_j \neq 0$, $j \in [k+1, n]$, is satisfied.*

Proof 2. *Suppose a single SET distorts the i^{th} residue digit in the residue vector of an integer X. It is represented as an addition of the number e_i to the residue while $0 < e_i < m_i$, namely,*

$$X' \longrightarrow [r_1, r_2 \cdots, (r_i + e_i) \cdots, r_{n-1}, r_n]$$

where X' is the altered number.
Therefore,

$$E = \sum_{j=1,j\neq i}^{n} a_j \prod_{l=1}^{j-1} m_l$$

such that

$$X' = X + \sum_{j=1,j\neq i}^{n} a_j \prod_{l=1}^{j-1} m_l$$

Since the redundant modulus m_{k+1} is larger than all the information moduli, E is larger than the legitimate range M in 1.3. The only case where the decoded integer appears in the legitimate range is $a_j = 0$, where $j \in [k+1, n]$. Consequently, $a_j \neq 0$ indicates the residue digit affection, otherwise, no single error occurs.

1.2.3.3 The single error correction

Theorem 3. *The RRNS codes with $(n-k)$ redundant moduli can detect $(n-k)$ errors and correct $(n-k)/2$ errors.*

This theorem has been demonstrated in different manners [YL73, BM73, BM74, KLS92, SK92]. The conventional RRNS codes that perform the error correction can be described in three steps:

1. Consistent checking:

   Detect all single errors with theorem 1. If no error is detected, then stop here, otherwise, go to the next step.

2. Erroneous digit locating:

   Find the location of the erroneous digit or digits. It is generally carried out by two approaches, mi-projections and based on BEX.

3. Recovery:

   Remove the corrupt digit or digits, and then restore the integer X from the rest error-free residues.

Example 1: An 8-bit RRNS codeword is generated from the modulo set (m_1, m_2, m_3, m_4, m_5)=(4, 5, 7, 9, 11), where m_4 and m_5 are redundant moduli. The legitimate range is $[0, 4 \times 5 \times 7) = [0, 140)$. The RRNS codeword for input data $X = 125$ is the residue vector $(1, 0, 6, 8, 4)$. Assuming that the third residue of the codeword is corrupted from 6 to 1 during the storing or the computation, these results in $X = (1, 0, 1, 8, 4)$. The following process based on the MRC, is applied,

1. consistent checking, discarding r_5, the recovered data is 1205 decoded by (r_1, r_2, r_3, r_4) and an error is detected.

2. erroneous digit is located, and iterations are executed,

3. m_i-projections are applied. Only one decoded integer is smaller than 140, as shown in the table below. So the correct integer $X = 125$ is restored.

Iteration	1	2	3
Discarded residue	r_1	r_2	r_3
Recovered data	3095	1709	125

Clearly, if the erroneous digit is r_4 or r_5, then the integer X can not be recovered until the discarding of the residue r_4 or r_5. Therefore, the m_i-projections need n cycles to obtain the error-free integer at the worst case.

1.3 Bi-Directional Redundancy Residue Number System

In the literature, Yau and Liu [YL73] introduced a method that uses appropriate computations to remove the error-correcting table. Mandelbaum [Man72] derived that two redundant moduli are necessary for single error correction. Then, Barsi and Maestrini [BM73] introduce a necessary and sufficient condition for the correction of a given error affecting that affects single residue digit of any legitimate number. They also proposed an RNS product code and further determined that if the error affects an arbitrary legitimate number or a number in overflow [BM74]. Jenkins et al. [EJ80, Jen83, BJ84, JA88] proposed residue number error checkers based on MRC

[Tay84] and implemented this method into digital filters thanks to efficient pipeline architectures. Their method disjoints one residue in each projection to process current error detection and diagnosis. A corrupt number is not actually corrected, but left out a series of m_i weighted projections. Ramachandran [Ram83] showed a particular procedure of modulo projections that discards more than one residue at each time. Su and Lo [SL90] used the redundant digits of MRC as the entries to build a lookup table. Krishna et al. [KLS92, SK92] bounded the RRNS with coding theory and derived the conditions for the single, double and multiple error corrections. Katti [Kat96] presented a method using a modulo set with common factors that leads to simplified error detection and correction.

The former methods detect or correct error using the computational iterations or an error-correcting table for the error detection and the error correction. However, this technique induces considerable hardware or delay overhead. In this work, we present a RRNS based fault-tolerant technique that demands no table to perform error-correction and also achieves fast error-free computation in the presence of a single SET. The proposed technique, Bi-directional Redundant Residue Number System (BRRNS) requires redundant moduli that satisfy some constraints to achieve fast error correction with minimum hardware overhead. The BRRNS is characterized by n pairwise relative prime positive moduli, m_1, $m_2 \cdots$, m_{n-1}, m_n, that are formed in increasing, i.e., $m_1 < m_2 < \cdot < m_{n-1} < m_n$. The first k moduli form a set of information, and the remaining $n-k$ moduli are redundancies. BRRNS code has the same structure as a RRNS code, but a BRRNS code requires that the redundant moduli satisfy the following conditions,

$$M = \prod_{i=1}^{k} m_i \approx M_R = \prod_{j=k+1}^{p} m_j, p \leq n, \tag{1.16}$$

$$M_n < m_1 \cdot m_2 \tag{1.17}$$

where the legitimate range of BRRNS is $Min(M, M_R)$ and any redundant modulus is smaller than the product of the two minimum redundant values as [Ram83]. As the product of certain redundant moduli is approximately equivalent to the product of the information moduli, any integer X belonging to the legitimate range can be also restored through the redundant residue vector, namely,

$$X \longrightarrow [r_{k+1} + r_{k+2} \cdots + r_{p-1} + r_p], p \leq n \tag{1.18}$$

Here, we define this property as alternative recovery.

Theorem 4. *The BRRNS codes with $n - k = 2$ redundant moduli is able to correct all single errors if the condition $m_1 < m_2 < \cdots < m_k < m_{k+1} < m_{k+2}$ is satisfied.*

By applying the theorem 3, this theorem has no need to be proved. However, the plural consistent-checking and the alternative recovery make the error-correction in the BRRNS codes faster than in the conventional RRNS codes.

With one redundant modulus is able to detect whether or not the single error occurs. Moreover, double consistent-checking can be done with two redundant moduli. The set of results obtained from double consistent-checking includes: one non-consistency and one consistency, two non-consistency and two consistency. The occurrence of single error is expected in two cases: the distortion of an information residue and the distortion of a redundant residue. In other words, the corrupt digit is either an information residue or a redundant residue and the spare digits are error-free. Consequently, with the property of alternative recovery, any integer X belonging to the legitimate range can be either restored from the information residue vector or from the redundant residue vector.

The BRRNS codes perform the error correction in two phases:

- **Phase 1: Plural Consistent-Checking**

  All single errors can be detected and located simultaneously.

- **Phase 2: Rehabilitation**

  The integer X belonging to the legitimate range is restored through the redundant residue vector or through the information residue vector according to the result of Phase 1.

Example 2: An 8-bit RRNS codeword is generated from the modulo set (m_1, m_2, m_3, m_4, m_5)=(4, 5, 7, 11, 13). The legitimate range is $[0, 4 \times 5 \times 7) = [0, 140)$, according to $Min(M, M_R)$. If an integer decimal message of $X = 125$ is considered, then the corresponding residue values are X=(1, 0, 6, 4, 8). Assuming an error occurs in r_3, then the received BRRNS codeword becomes (1, 0, 1, 4, 8),

1. **Plural consistent-checking**

   r_1, r_2, r_3 and r_4 involved in the first consistent checking, then, a_4=5≠0, erroneous. r_1, r_2, r_3 and r_5 involved in the first second checking, then, a_5=7≠0, erroneous. Consistent-checking are faulty, hence, the corrupt digits are in the information residue vector.

Serial Process	1	2	3	4	5
Radix	a_1	a_2	a_3	a_4	a_5
Value	1	1	4	5	7

2. **Rehabilitation**

   The integer X is restored by r_4 and r_5, the redundant residue set.

$$a'_4 = r_4,\ a'_5 = (r_5 - a'_4) \times m_{4,5}^{-1}(mod\ m_5),$$

	a'_4	a'_5	X
Value	4	11	125

1.4 Architecture for BRRNS Decoder

In this section, we present architectures to implement pipeline BRRNS decoder. At first, the architecture of a RRNS based carry-save adder that requires five clock cycles at worst to correct single error is recalled. The self-diagnosis BRRNS decoder based carry-save adder that is able to instantly correct single error is illustrated after.

1.4.1 Conventional RRNS error checker

In the literature [EJ80,Jen83,BJ84,JA88], Jenkins and his colleagues proposed residue number error checkers based on the RRNS. They also implemented this method into digital filters based on an efficient pipeline architecture. For instance, if the conventional RRNS code with $k = 3$ and $n = 5$ is considered, then the mixed radix digits a_i for each of n projections required for error decoding are computed by the expressions of MRC given in (1.10), namely,

$$a_1 = r_1,$$

$$a_2 = (r_2 - a_1) \cdot m_{1,2}^{-1}(mod\ m_2),$$

$$a_3 = ((r_3 - a_1) \cdot m_{1,3}^{-1} - a_2) \cdot m_{2,3}^{-1}(mod\ m_3),$$

$$a_4 = (((r_4 - a_1) \cdot m_{1,4}^{-1} - a_2) \cdot m_{2,4}^{-1} - a_3) \cdot m_{3,4}^{-1}(mod\ m_4),$$

$$a_5 = ((((r_5 - a_1) \cdot m_{1,5}^{-1} - a_2) \cdot m_{2,5}^{-1} - a_3) \cdot m_{3,5}^{-1} - a_4) \cdot m_{4,5}^{-1}(mod\ m_5) \tag{1.19}$$

These computations are performed in a serial way, thus, a pipelined architecture for the implementation of the mixed radix converter is mostly employed in digital filters, as shown in Fig. 1.5. For the sake of clarity, the Look-up tables formed as ROM take place of the computations for the radixes conversions. Thus, the residues or the former radixes are read as the address to fetch the radix. In the case of five-moduli, the pipeline takes five execution cycles until a_5 fan-outed, which is one cycle of m_i-projections. In practice, one of the five residues is dumped during each iteration. Thus, the capability of performing the consistent-checking (see also Section. 1.2).

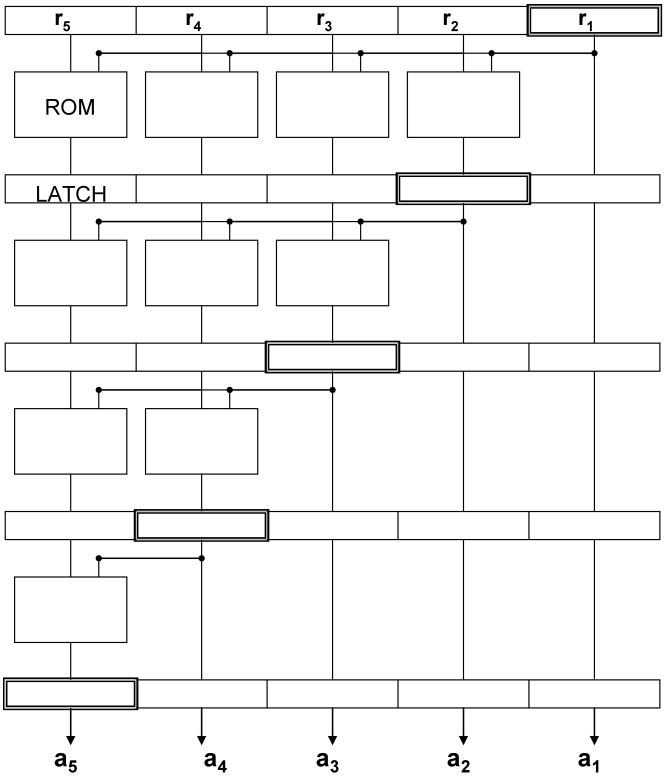


Figure 1.5: Error checker based on the MRC for a systematic RRNS code with $k = 3$ and $n = 5$.

However, the consistent-checking performed during the iterations enables some detections but no correction. Actually, once single soft-error occurred, the m_i-projections requires n iterations at the worst case to obtain an error-free number. For this reason, it mostly impacts the real-time decoding process achieved with a low latency. The digram of a carry-save adder based on RRNS that is able to correct single error occurred during the addition process or the correction process is shown in Fig. 1.6.

For instance, the occurrence of a single SET is considered. Due to the affection of the SET, it turns out that one of the five residues r_1, r_2, r_3, r_4, and r_5, is erroneous. Suppose that r_1 is false. In this case, in order to recover an error-free digit X, the RRRNS based carry-save adder performs an iterative error-correction process as shown as the diagram presented in Fig. 1.6. At first, four residues, r_1, r_2, r_3, and r_4, are chosen to perform an error-detection, by checking the value of generated a_4 through (1.19). Since the false residue is r_1, thus, a_4 is not equal 0. Then a second error-detection with r_1, r_2, r_3, and r_5, is started. By checking the value of generated a_5 through (1.19), due to the corrupted r_1 a_5 is not equal to 0 neither. Similarly, a third error-detections via r_1, r_2, r_4, and r_5 and a fourth error-detections via r_1, r_3, r_4, and r_5 are started. Both checkings end up by $a_5 \neq 0$ due to the affection of r_1. At the firth error-detection, r_2,

r_3, r_4, and r_5 are picked up to check the value of a_5. Since all the four residues are correct, it passes through the error-checking and also forward the *Shift and Addition* block three radix digits a_2, a_3, and a_4 to recover the digit X.

Actually, it requires five execution cycles at worst to correct a single error. In the following subsection, the self-diagnosis decoder based on the BRRNS for single-error is presented.

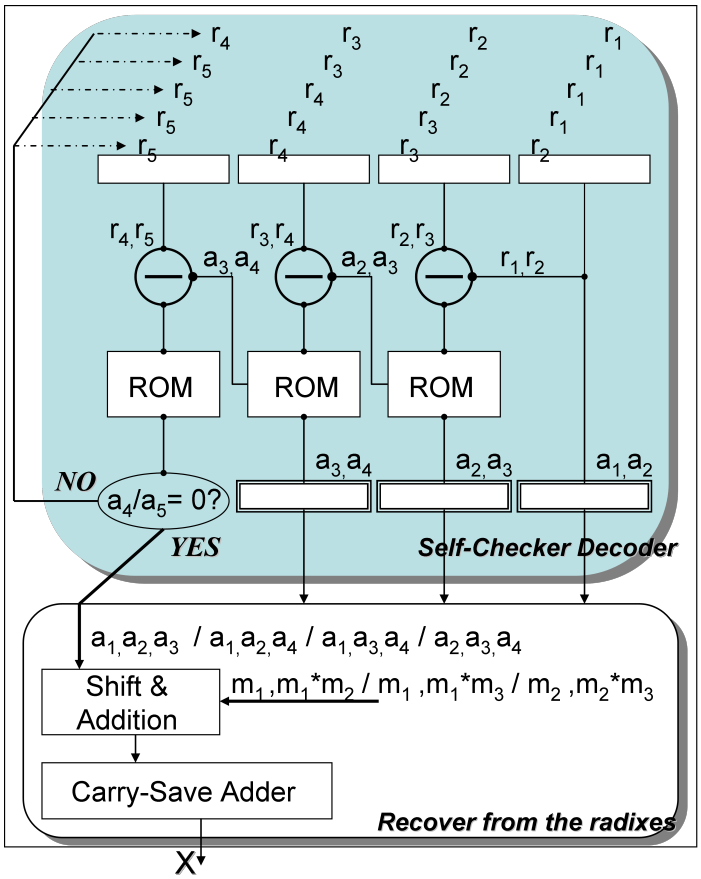


Figure 1.6: RRNS based carry-save adder diagram can correct single error occurred during the addition or the correction. However, it needs five execution cycles at worst.

1.4.2 BRRNS self-diagnosis decoder

If the same context is considered, a BRRNS code with $k = 3$ and $n = 5$ has a different structure for the computations of the mixed radix digits a_i.

$$a_1 = r_1,$$

$$a_2 = (r_2 - a_1) \cdot m_{1,2}^{-1} (mod\ m_2),$$

$$a_3 = ((r_3 - a_1) \cdot m_{1,3}^{-1} - a_2) \cdot m_{2,3}^{-1} (mod\ m_3),$$

$$a_4 = (((r_4 - a_1) \cdot m_{1,4}^{-1} - a_2) \cdot m_{2,4}^{-1} - a_3) \cdot m_{3,4}^{-1} (mod\ m_4),$$

$$a'_5 = (((r_5 - a_1) \cdot m_{1,5}^{-1} - a_2) \cdot m_{2,5}^{-1} - a_3) \cdot m_{3,5}^{-1} (mod\ m_5) \tag{1.20}$$

As explained in section 1.3, the proposed method performs the plural consistent-checking in parallel. By observing a_4 and the one by observing a'_5, the consistent-checking are processed during the same cycle. Therefore, here, a pipeline architecture based on this principle for single-error correction can be achieved as shown in Fig.1.7.

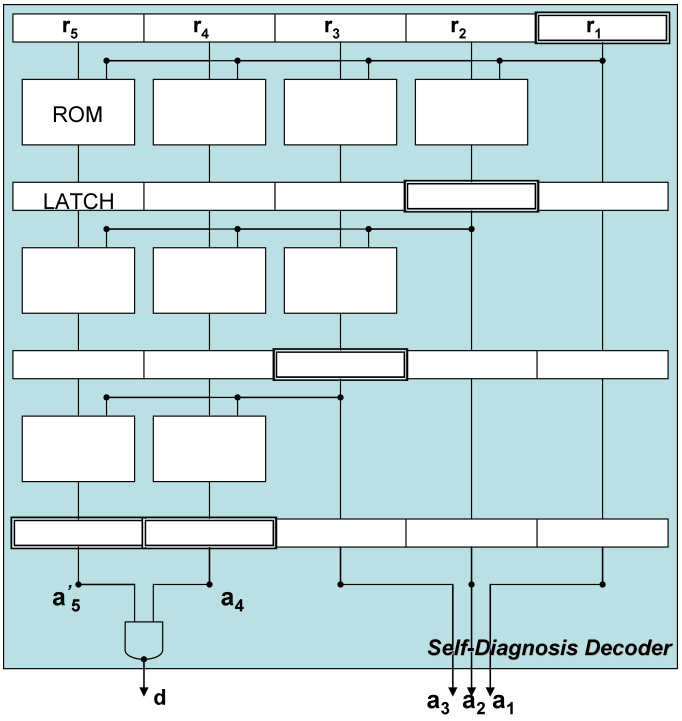


Figure 1.7: Self-Diagnosis decoder for single-error correction based on the MRC for a systematic BRRNS code with $k = 3$ and $n = 5$.

During the first three execution cycles, a_1, a_2, and a_3 values are computed, respectively. Afterward a_4 and a'_5 values are obtained after the fourth cycle with equation 1.20. To determine the set for the data recovery, an AND logic gate is employed to observe the value of a_4 and a'_5. Moreover, "signal $'d'$=0" indicates that the information residues, a_1, a_2, and a_3, are error-free. Otherwise, a_4 and a'_5 are error-free while "d=1". Subsequently, after the self-diagnosis decoder, the addition based on either a_1, a_2 and a_3, or a_4 and a'_5 values is required for completing the rehabilitation of the integer X.

A BRRNS based carry-save adder diagram is shown in Fig.1.8. This architecture can correct single error during the addition process or the correction process during a single

execution cycle. More precisely, any single error event occurs among the residues, r_1, r_2, r_3, r_4, and r_5, or during the consistent-checking, or during the recovering data process, can be masked. For instance, if an error event affects the self-diagnosis process, it can be detected via the consistent-checking. Through the detecting, $a_4 = 0$ and $a_5 = 0$, a decision is made to recover the integer X by either r_5 and r_4, or by a_1, a_2, and a_3, as shown at the bottom of Fig. 1.8.

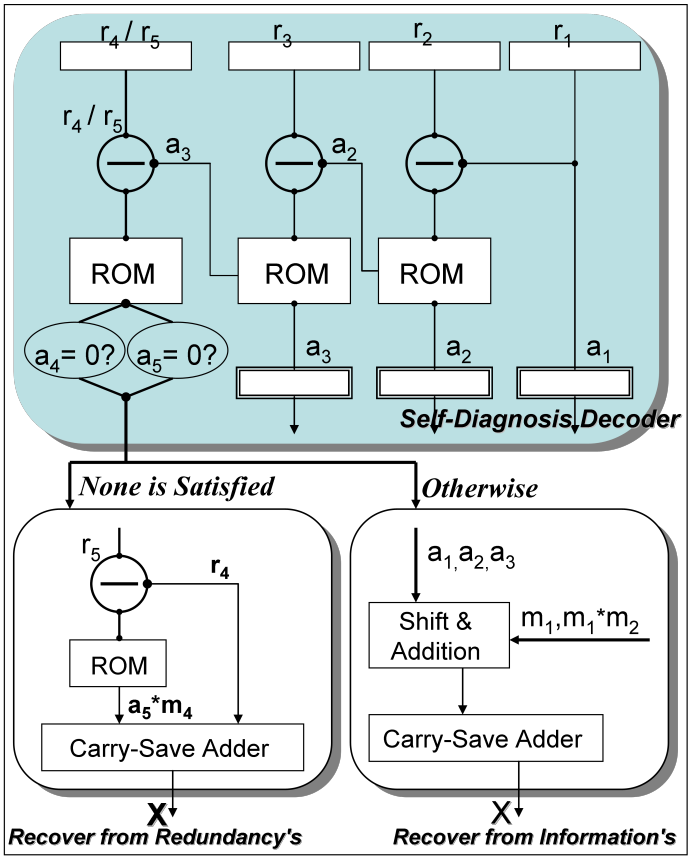


Figure 1.8: BRRNS based carry-save adder diagram, instantly correcting single error during the addition or the correction.

1.5 Conclusion

In this chapter, we presented a new single-error correction method based on residue number arithmetic, as a contribution to the embedded arithmetic error-correction to the unreliable circuit.

At first, same related works are reviewed: traditional fault-tolerant method, Triple Modular Redundancy (TMR) and Algorithmic Noise-Tolerance (ANT) based methods. The TMR is a well-known fault-tolerant method, which is able to correct single SET

by charging double modular overhead to the design. However, the clumsy redundancy and malfunction when internal transient faults occur confine TMR to be wide adapted as a good fault-tolerant method. Algorithmic Noise-Tolerance (ANT) can be regarded as one of the dual modular redundancy family, which duplicates modular function once and makes a decision from the original function and the replica to prevent SET occurrence. Since the replica in ANT technique may be attacked by SEU, Algorithmic Soft-Error Tolerance (ASET), based on ANT, employs a second replica to enforce the error-resilience. Nevertheless, two drawbacks limit the based ANT method to be widely applied indeed, it is hardly adapted while the reduced precision based estimator can not be realized in a feasible way. Moreover, it requires that the decision block is reliable so as to perform a correct error-control. For another robust model, Error Resilient System Architecture (ERSA) has been addressed for low-cost robust system. In order to achieve low cost, this robust model uses the asymmetric reliability architectures: highly reliable and expensive cost for the main thread, and inexpensive nor unreliable worker threads. However, it takes a higher costs for general-purpose applications to adapt this model.

Secondly, we recalled the fundamentals of a Residue Number System (RNS) and of a Redundancy Residue Number System (RRNS). After, a novel method based on RNS called Bidirectional Redundant Number System (BRRNS) [TBJJ10] is proposed and its proofs relevant to single-error detection and correction were detailed. Finally, a pipeline hardware architecture for the BRRNS was described for the single-error correction. Moreover, the architecture for BRRNS based carry-save adder to waive single error from the addition process or the recovering process is also proposed.

Two properties are benefited by using the BRRNS. Thanks to the plural parallel consistent-checking that performs the error detection and the error diagnosis, fast error correction can be achieved. As an arithmetic fault-tolerant technique, any SET occurs anywhere within the BRRNS can be precluded.

For a future directions, we will turn our attention to extend a BRRNS into a double-error correction issue. In addition, it would be interesting to see the applications of a BRRNS code as well as their performances. To build an inherent error-resilient BRRNS may be grate useful for the future robust computation design.

2

Criteria for a Good Embedded Decoder

To evaluate the reliability and the hardware efficiency of a robust design on unreliable circuit, we present a new dual-space criteria in this chapter. The novel criteria, namely Reliability-Efficiency Criteria (RE-Criteria), defines a two dimensional solution space, which consists of: error probability at fanout and hardware efficiency estimations. In addition, an example of Pareto distribution is given. This distribution is based on a classical FIR filter performed with automatic repeat request, TMR error-correction and BRRNS error-correction mechanisms based architectures is given.

In these criteria, taking into account that the correction unit is also subject to error, we propose to assess the quality of an architecture using not only its efficiency (i.e. the normalized number of operation per area and per unit of time), but also its final output error rate. By using the proposed dual space criteria, a two dimensional space results, namely a Pareto distribution can be observed. Via the Pareto curves, designers are able to discern the properties, the reliability and the efficiency, of fault-tolerant strategies.

Since researchers have been laying stress on robust system design, the requirement of a good criteria to evaluate a good fault-tolerant technique is forthcoming. In fact, an accurate error model for unreliable circuit is complicated. Two main issues, high hardware complexity and large size fan-in and fan-out, hamper the modeling. In this work, we propose a very simple error model as a worst-case metric. This error model only requires the hardware area and time execution of a design. Moreover, the proposed hardware efficiency metric, only operates on the hardware area and time execution as well. Therefore, our reliability-efficiency criteria would be widely adapted for the purpose of evaluation on any robust design.

2.1 Introduction

Currently a number of approaches have been proposed to cope with the reliability analysis issue. To evaluate the SER, the obvious method is to inject some errors into the architecture and then simulate the design thanks to input vectors. A very first method is Built-In Self-Test (BIST) [AKS93], that consists of pattern generator and response analyzer. The response analyzer compares fanout from the testing design with the result according to the input from the pattern generator. However, the timing issue

restricts BIST to a good feasible test method. Moreover, the real-life test [ARB+07] that exposes a circuit under the radiation beam to obtain the Soft-Error Rate (SER) also requires large number of chip samples and is costly in terms of time. Under this circumstance, the transient fault based model to evaluate the SER resulting from a transient fault propagation through a design provides a feasible way for a reliability analysis.

In the literature, several models have been presented to achieve efficient estimations of transient faults with a small estimation error threshold. However, due to the circuit tending towards rather high complexity and large size fan-in, the present formal reliability analysis methods, namely analytical and symbolic methods [MZM10], are hardly tackled with circuit evaluation. Consequently, an alternative approach, based on a simple and pessimistic error-model with adequate reliability estimation ability, is more appropriate. On the side of SE mitigation, a considerable amount of approaches has been proposed to mask or mitigate SE in different contexts. While those approaches laid emphasis on hardware complexity, power consumption, and delay overhead. Note that in most cases, the functionality for detecting or correcting error is usually regarded as error-free. But this assumption does not conform to the trend. The need of formulating some metrics for the fault-tolerant and efficient featured design in undependable devices is forthcoming.

In this chapter, taking into account that the correction mechanisms are also subject to error, we present a new metric, termed as *Reliability-Efficiency Criteria* (RE-Criteria). This metric helps the design in unreliable computation. According to the requirement of the application, the **Pareto** distribution enables to select the best compromise between hardware efficiency and resource reliability. This approach characterizes an architecture in unreliable hardware by a two dimensional criteria: the Reliability (probability of non error in the output of the architecture) and the Efficiency of an architecture. From those two criteria, it is possible to plot a Pareto distribution for several architectures with different error detection and correction mechanisms.

In the proposed error probability metric, the model of transient error is simple. If a circuit of normalized area S has a probability p of generating a correct result during one clock cycle, a computation requiring a normalized area of $n.S$ during m clock cycles has a probability $p^{n.m}$ to product a good result. Note that the metric implies that all SETs impact the output. But, it is not necessarily the case. Thus, the proposed metric is a **worst-case metric**. Indeed, this metric guarantees a level of output confidence. In this chapter, after reviewing well-known correcting techniques with the RE-Criteria, we detail an example of Pareto distribution based on a classical FIR filter performed with the error-correcting architectures.

2.2 The Reliable Efficiency Criteria

2.2.1 The unreliable computing model

A generic computing system with its capacity of correction is described in Fig.2.1. It is composed of an encoder, a computing function, and a decoder. The input of the encoder is assumed error-free. The encoder has to perform any coding before the computation. For example, triplication by wires or more complex coding involving computation is also subject to SET. The computing function performs the required computation in the adequate information representation. The decoder generates the output in the correct format after eventually the detection and/or the correction of errors due to SET. Although related work that considers the decoder as error-free, in our study, the decoder can also be affected by a SET (see Fig.2.1-(a), classical model and Fig.2.1-(b), the proposition).

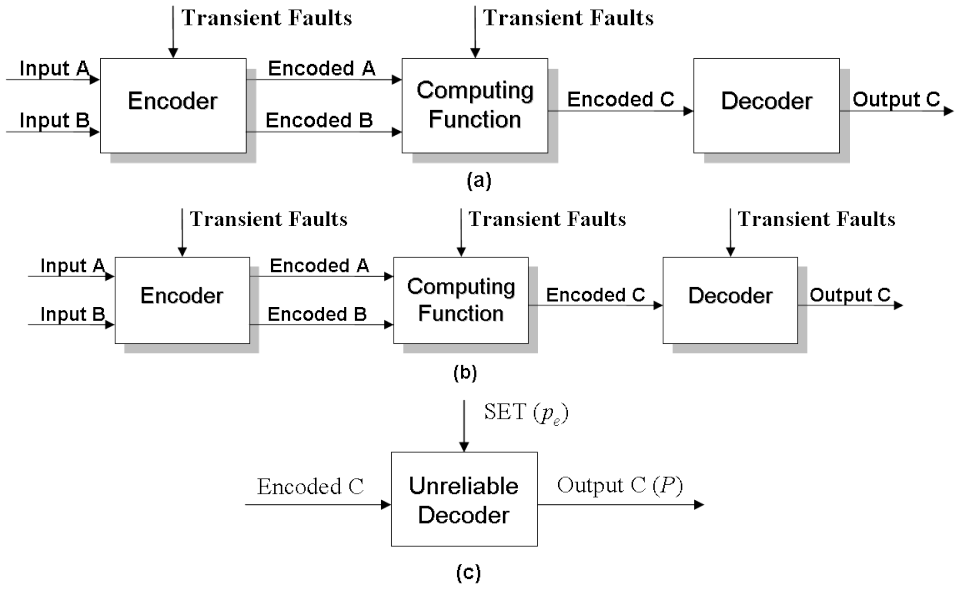


Figure 2.1: a. Block diagram of a conventional computing system, b. Block diagram of an unreliable computing function system, c. A decoder diagram from unreliable computing system.

2.2.2 The model of reliability

We consider that the "unitary" probability of SET occurrence in the decoder is a constant for a given CMOS processes, supply voltages and external environment of the device, such as ionization level, temperature, etc [NBS+07, GSN+07, Bau01]. This "unitary" SET probability p_e denotes the probability of error in a unity area during one

clock cycle. On the contrary, $\bar{p}_e$, denotes the probability of error-free computation in a unity area during one clock cycle:

$$\bar{p}_e = (1 - p_e) \tag{2.1}$$

The error probability of the output is expressed as P, as shown in Fig.2.1-(c), and the resulting no-error probability $\bar{P}$. In order to upper bound the value of P, three hypotheses are assumed:

1. *Isotropy*: the unitary probability of SET is constant for all the designs and independent of the underneath logic.
2. *Contamination*: If an error occurs in a design, or in one of its inputs, its output will be corrupted.
3. *Irreversibility*: Two successive errors cannot lead to a correct result, the final result will stay erroneous.

Those three hypotheses will be applied only on computational logic. For decoding task with error correction mechanism, we will derive more precise model.

One should note that the hypotheses are all pessimistic and correspond to a worst-case configuration. In fact, the error probability of a "unitary" area should depend on its logic. Thus, *Isotropy* is an approximation. Moreover, it is easy to find an example where *Contamination* is false. Thus, if a NAND logic gate has one input that is equal to zero, an error in the other input is not propagated. If two errors occur consecutively on a binary signal the result will be corrected, which again, is in contradiction with the *Irreversibility*. Finally, when dedicated error correcting hardware is used, *Contamination* and *Irreversibility* have not to be taken into account.

Nevertheless, those three hypotheses allow us to build a worst-case result, i.e., to guarantee a given probability of error for the output of the function. Although the definition of a more precise model is out of the scope of this study, the explicit computation of error probability in output will be derived in the following.

Let us study how the error probability scales on a more complex design under the three hypotheses. Fig.2.2 shows two configurations that both require two area units. In the first case, the parallel one (Fig.2.2-(a)), the resulting output is correct if the outputs of the two components are correct, i.e. the probability for error-free output is then $\bar{P} = \bar{p}_e^2$. In the second case, the serial one (Fig.2.2-(b)), since an error in f_1 affects the output by *Contamination* and an error in f_2 also affects the output by Irreversibility, then both computations should be correct, which leads again to, $\bar{P} = \bar{p}_e^2$. This formula can be generalized to a design of any size n in terms of unit area. Moreover, if the same architecture is used during several clock cycles m to process an output, all cycles should be error-free. The probability of correct computation is then given by

$$\bar{P} = (1 - p_e)^{n \cdot m} \tag{2.2}$$

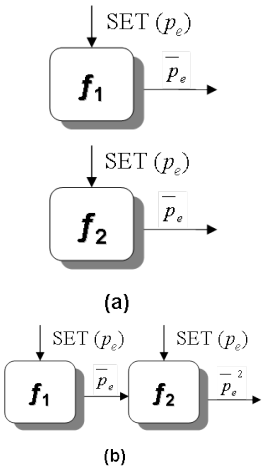


Figure 2.2: a. Parallel functions, b. Serial functions.

2.2.3 The RE-Criteria

A complexity-throughput metric observing the efficiency of an architecture has been proposed for Forward Error Correction (FEC) decoding in [Mul07]. It defines the hardware efficiency of an architecture as the normalized number of operation per area unit and time unit. Note that the nature of the operation is not specified, it can be an FFT transformation, a decoding iteration in an iterative decoder, a multiplier or anything else. Assuming that an architecture executes a function with a component of n area units and during m clock cycles, its efficiency is defined as,

$$\gamma = \frac{1}{n \cdot m}, operation/(area\ unit \times time\ unit) \tag{2.3}$$

Further, from 2.2, the error probability P in output can be expressed as,

$$P = 1 - \bar{P} = 1 - (1 - p_e)^{n \cdot m} \tag{2.4}$$

Consequently, the proposed RE-Criteria consist of a couple (P,γ) of metrics that are the metric for the error probability and the metric for the hardware efficiency as, expressed in 2.4 and 2.3, respectively. From those two estimations, we are able to plot a Pareto distribution, that characterizes the tradeoffs between the efficiency and the reliability among the architectures. The formal derivations of the RE-Criteria for different state of the art correcting strategies are presented in the next section.

2.3 Formal Derivations of the RE-Criteria

This section presents different conventional strategies to detect and/or correct errors. Considering single-error correction, error probabilities based on the metric are detailed

for each case.

2.3.1 Spatial-TMR

A conventional solution duplicates the functional unit. Basically, to apply TMR [LV62], three identical units are designed to execute the function from the same input data. The three output results are sent into a majority voter that allows us to correct at most one error (Fig.2.3-(a)). In summary, the output of the Spatial-TMR (S-TMR) is wrong when SETs induce at least two faulty modules (event of probability $P_{>1}$) or when there is one faulty module and the voter is faulty (event of probability Q). The resulting error probability of S-TMR decoder P_{S-TMR} can be expressed under the three hypotheses of *Isotropy*, *Contamination*, and *Irreversibility*, as follows:

$$P_{S-TMR} = P_{>1} + Q \cdot \bar{P}_{>1} \tag{2.5}$$

Let P_R be the error probability in a single module of size n that performs a computation in m cycles, as introduced in the previous section, then,

$$P_R = 1 - (1 - p_e)^{n \cdot m}$$

$P_{>1}$ is expressed as:

$$P_{>1} = \binom{2}{3} \cdot P_R^2 \cdot \bar{P}_R + P_R^3$$

Moreover, let n_Q be the area cost of the voter that is used during one single clock cycle after the computation, thus,

$$Q = 1 - (1 - p_e)^{n_Q}$$

Thanks to the triplication associated with voter, the efficiency of S-TMR can be reduced to:

$$\gamma_{S-TMR} = \frac{1}{3 \cdot n + n_Q} \cdot m \tag{2.6}$$

Therefore, by using RE-Criteria, the S-TMR solution corresponds to a single point (P_{S-TMR}, γ_{S-TMR}) in the Reliability-Efficiency two dimensional space, termed as RE-Space.

2.3.2 Temporal-TMR

If the same instance is considered, an alternative solution is to reuse the computing function, as shown in Fig.2.3-(b). Compared with S-TMR, temporal-TMR (T-TMR) [SS06] is regarded as a tradeoff between area and time. The resulting error probability for the T-TMR holds the same form as the S-TMR, P_{T-TMR}

$$P_{T-TMR} = P_{>1} + Q \cdot \bar{P}_{>1} \tag{2.7}$$

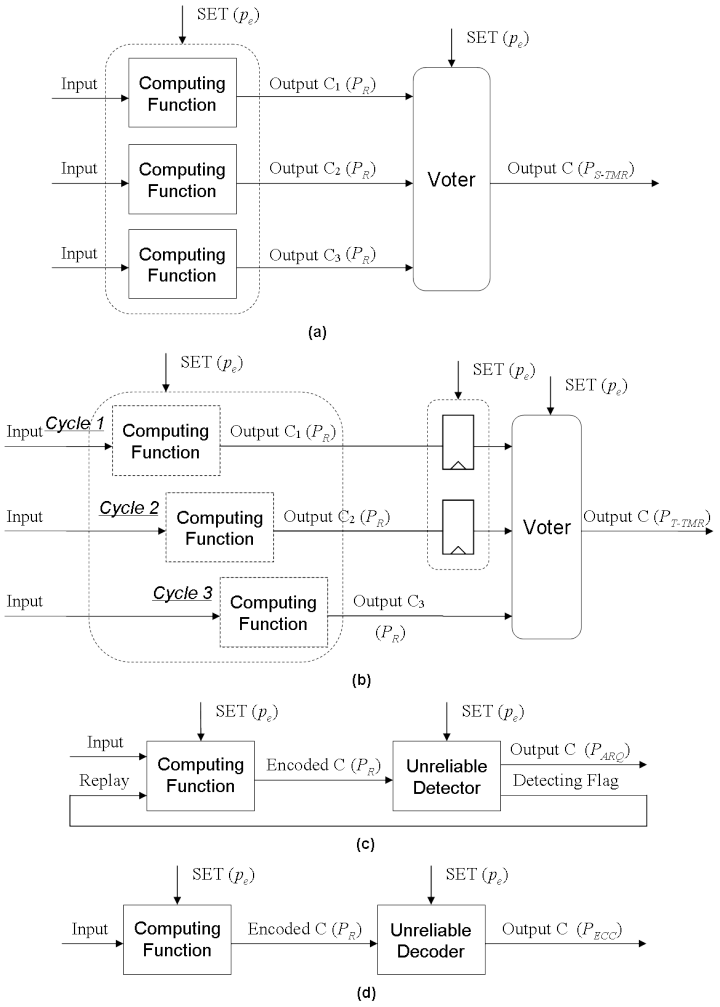


Figure 2.3: a. S-TMR structure; b. T-TMR structure; c. ARQ structure; d. Error Control code structure.

However, the expression of $P_{>1}$ should be adapted because SEs can occur in the registers that contain the results of first computations during $2.m$ clock cycles or the results of the second computations during m clock cycles. Let $\bar{R}_1$, $\bar{R}_2$ and $\bar{R}_3$ denote the probability of correct input into the voter in the first, the second, and the third cycle, respectively. More precisely, suppose n_R the area cost of each register, then $\bar{R}_1 = \bar{P}_R \cdot (1-p_e)^{2m \cdot n_R}$, $\bar{R}_2 = \bar{P}_R \cdot 1 - {p_e}^{m \cdot n_R}$ and $\bar{R}_3 = \bar{P}_R$. $P_{>1}$ is thus expressed as:

$$P_{>1} = \sum_{i_1,i_2,i_3 \in (1,2,3)}^{i_1 \neq i_2 \neq i_3} R_{i_1} \cdot R_{i_2} \cdot \bar{R}_{i_3} + \prod_{i=1}^{3} R_i^3$$

For the efficiency of the T-TMR, the clock cycle is tripled by reusing the function

temporally,

$$\gamma_{T-TMR} = \frac{1}{3m \cdot n + n_Q + n_R} \tag{2.8}$$

As a register area cost is low, its error probability is much smaller than P_R. Hence, P_{T-TMR} is approximate to P_{S-TMR}, as well as its hardware efficiency.

2.3.3 ARQ

The Automatic Repeat reQuest (ARQ) [LCM84] technique performs the error-detection by adding some redundancy. When an error occurs, the detector asks to restart the computation. The data is sent once it is checked as error-free, as shown in Fig.2.3-(c).

Generally, the output of the redundant computation includes three cases: error-free, error-detectable, and error-undetectable. Error-detectable output (probability P_{R-1}) is an erroneous output and is not a codeword. In this case, the incorrect output can be found by the detector. Otherwise, the output that turns into one of the codewords is termed as error-undetectable output (probability P_{R-2}). Consider an error-detecting code (c_n, c_k, c_{n-k}). When a single error occurs, P_{R-1} can be expressed as, $P_{R-1} = \frac{2^{c_n}-2^{c_k}}{2^{c_n}-1} \cdot P_R$. Thus, P_{R-2} is derived, $P_{R-2} = P_R - P_{R-1}$.

Let us consider the cases for resulting data from the ARQ engine. At first, if the computational process is error-free, namely the output from the computation is correct, the computation is restarted when the detector is faulty. Otherwise, the data is correct. Secondly, in case of error-detectable output, the data is incorrect when the detector is faulty. If not, the computation is restarted when the detector operates correctly. At last, in the case of error-undetectable, the detector gives erroneous output when it runs correctly. If not, when it is faulty, the detector asks to restart the computation. Assuming that all the inputs that are required in the replay are correct. Consequently, the state of data is summarized into three events: correct output (probability P_{ARQ-C}), incorrect output (probability P_{ARQ-E}), and restart computation (probability P_{Replay}). From the case analysis, probabilities of the three events can be derived

$$P_{ARQ-C} = \bar{P_R} \cdot \bar{T} \tag{2.9}$$

$$P_{ARQ-E} = (P_{R-1} \cdot T) + (P_{R-2} \cdot \bar{T}) \tag{2.10}$$

and

$$P_{Replay} = \bar{P_R} \cdot T + P_{R-1} \cdot \bar{T} + P_{R-2} \cdot T \tag{2.11}$$

where T is the error probability of the detector. Moreover, Fig. 2.4 illustrates the detecting diagram for a k-time ARQ.

Finally error probability of a k-time ARQ P_{ARQ} can be obtained by an accumulation of the P_{ARQ-E} from the first-time ARQ till the k-time ARQ. In this case, P_{ARQ} is

expressed as

$$P_{ARQ} = \sum_{\infty}^{k=0} (P_{R-1} \cdot T + P_{R-2}) \cdot P_{Replay}^{k} \tag{2.12}$$

If the detector takes n_T unit area and needs m_T clock cycles for each detection process, then $T = 1 - (1 - p_e)^{n_T \cdot m_T}$, and

Moreover, the average number of cycle Ns for the k-time ARQ is

$$Ns = \sum_{k=0}^{\infty}(k+1) \cdot (n + n_T) \cdot P_{Replay}^{k} \cdot \bar{P}_{Replay},$$

It can be simplified as:

$$Ns = \frac{(n+n_T)}{\bar{P}_{Replay}}.$$

Hence, the efficiency for the k-time ARQ is expressed as:

$$\gamma_{ARQ} = \frac{\bar{P}_{Replay}}{(n + n_T) \cdot (m + m_T)} \tag{2.13}$$

We recall that for a sake of the redundancy engaging, n and m that are used to compute the error probability P_R are different from the original function. More precisely, the error probability rapidly decreases with an increase of k. Meanwhile, it degrades the latency and data rate. Consequently, the compromise lies within RE-Space (P_{ARQ}, γ_{ARQ}).

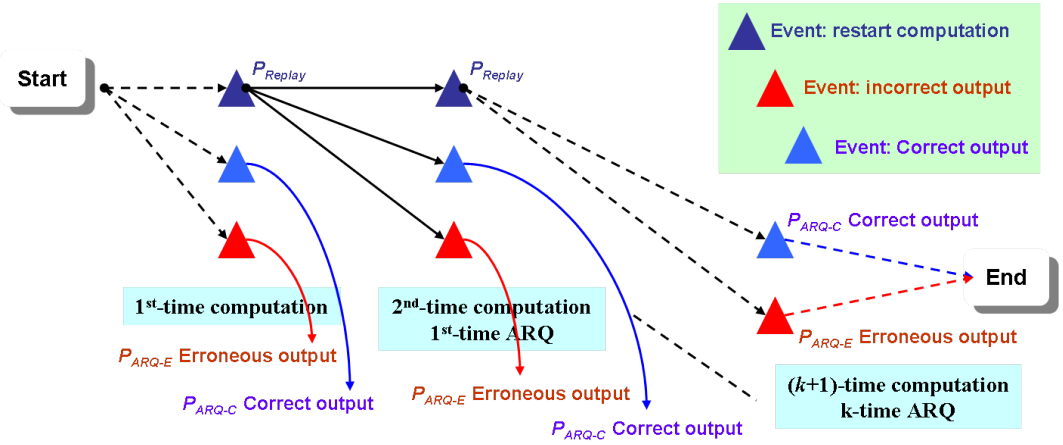


Figure 2.4: The detection diagram for K-time ARQ.

2.3.4 Error Control Code

Instead of restarting the computation, a decoder can perform both the detection and the correction, known as the error control code, as described in Fig.2.3. This technique

is an extension of the S-TMR technique that uses a repetition error control code (3, 1, 2) to find a single error. The output of the computation includes three cases, error-correctable, error-uncorrectable, and error-free. Thus, the data is erroneous when the output of the computing function is error-correctable and the decoder is also faulty. Moreover, the data is also erroneous when the output of the computing function is error-uncorrectable. Hence, the error probability of the error control code decoding based computation P_{ECC} is

$$P_{ECC} = P_{Rc} \cdot D + P_{Ru} \tag{2.14}$$

where D denotes the error probability of the decoder and P_{Rc} and P_{Ru} represent the probability of error-correctable and error-uncorrectable, respectively. Let n_D unit area and m_D clock cycles for the decoder, then $D = 1 - (1 - p_e)^{n_D \cdot m_D}$. Obviously, P_{Rc} and P_{Ru} depend on the decoding algorithms.

Let us consider a case of BRRNS based error correction as described in Section. 1.3. In general, the restoration process is the self-diagnosis decoder as shown in Fig. 1.4.2. The self-diagnosis decoder takes the residues that are produced from the digit-to-residue blocks. If the number of modulo for BRRNS code is 5, the error probability of each digit-to-residue block to generate a corresponding residue is P_{resi}, Then the erroneous data from the decoder corresponds to two cases. At first, more than two incorrect residues are obtained from the digit-to-residue blocks (probability P_{Ru}), as the error-uncorrectable case. Second, the case of correctable error. No more than two digit-to-residue block is faulty (probability P_{Rc}) while the decoder is faulty.

Thus, P_{Rc} and P_{Ru} from (2.14) are summarized as

$$P_{Rc} = 1 - P_{Ru} = (1 - P_{resi})^5 + \binom{2}{3} \cdot (1 - P_{resi})^4 \cdot P_{resi} \tag{2.15}$$

where $P_{resi} = 1 - (1 - p_e)^{n_re \cdot m_re}$, and n_{re} and m_{re} represent the number of area unit and the number of time unit for each digit-to-residue block, respectively.

Moreover, the efficiency of the error control code is:

$$\gamma_{ECC} = \frac{1}{(n + n_D) \cdot (m + m_D)} \tag{2.16}$$

Naturally, a more complex coding scheme can be applied to increase the efficiency/error-probability capability of the code, i.e. iterative decoding. To sum up, (P,γ) RE-Space that is obtained by using the RE-Criteria, is able to illustrate the tuning among those parameters (reliability, efficiency, latency, data rate, etc...) for the requirements.

2.4 Pareto Distribution

In this section, the RE-Spaces (P, γ) draw out the Pareto distribution for different strategies that were previously detailed. These distributions illustrate the trade-offs

between the hardware efficiency and the transient error probability.

2.4.1 The experimental results

The objective of the experimental investigations is to demonstrate the interests of Pareto distribution based on the RE-Criteria. Therefore, a well-known pipeline FIR filter is considered as a typical case study. Let $x(n)$ be the input integer of the FIR filter and $y(n)$ be the output integer,

$$y(n) = \sum_{i=0}^{N} x(n-i) \tag{2.17}$$

In order to perform the error-correction in the filter architecture, the TMR technique that is carried out by duplicating the modules in spatial or in temporal has been first applied as shown in Fig.2.5-(a). Moreover, the ARQ technique applied to FIR filter is able to restart the computation by using a detector, see Fig.2.5-(b). We also turn our attention to FIR filters using RNS [Gar59] arithmetic so as to implement error control code based correction. The RNS implementation of a FIR filter is decomposed into k parallel filters [Con08]. By adding some redundant residues, the RNS that has the error detection and correction properties is called Redundant RNS (RRNS). Moreover, the technique, called Bidirectional RRNS (BRRNS) [TBJJ10], that requires redundant moduli for satisfying some constraints to achieve fast error correction, can be regarded as an error control code. A FIR filter that is implemented in RNS arithmetic, with applying BRRNS self-diagnosis decoder, is shown in Fig.2.5-(c).

Table. 2.1 details the logic synthesis results of size various FIR filters in terms of number of slice (the elementary programmable logic block in FPGAs) and clock frequency. In this work, four FIR filters have been investigated for different order N and different input data bit length l. Each filter has been designed for each of the four strategies. Note that the simplex denotes the filters where no correcting mechanism is applied. The error correcting techniques implemented are: the FEC kind, the S-TMR and the BRRNS based, the ARQ kind, modulo-4 check, modulo-8 check, modulo-16 check and the spatial Double Module Redundancy (DMR). The proposed criteria are under *Isotropy* hypothesis. It means that SET probability p_e is constant (see Section.2.2.2). For a sake of facility, p_e is settled to 10^{-13} (equivalent to the value of one FIT). By using 2.5-2.16, the error probability and the efficiency of the filters are estimated. Moreover, both the values of error probability and hardware efficiency for different filters are normalized by the ones of the simplex considered as the reference. Since the error probability of T-TMR is estimated approximately as the S-TMR, its results are skipped.

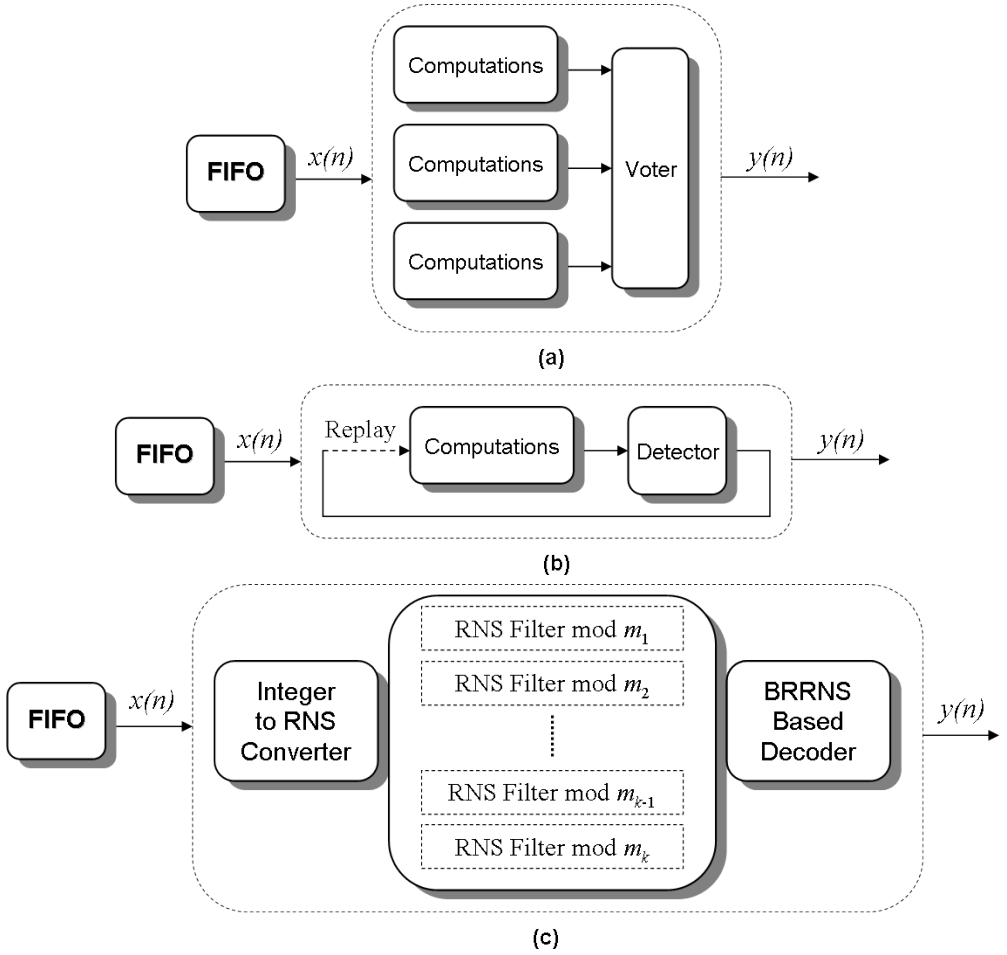


Figure 2.5: a. S-TMR based FIR filter structure; b. ARQ based self-detecting FIR filter structure; c. FIR filter implemented in RNS arithmetic, with also a BRRNS self-diagnosis decoder.

FIR Filter Sizes (N, l)	FIR-I (16, 5)	FIR-II (32, 6)	FIR-III (64, 7)	FIR-IV (128, 8)
Strategies	Slice/Fre.	Slice/Fre.	Slice/Fre.	Slice/Fre.
The Simplex	[94, 424]	[128, 417]	[171, 410]	[232, 403]
S-TMR	[249, 405]	[383, 398]	[508, 300]	[681, 309]
BRRNS Based	[477, 353]	[843, 263]	[1133,232]	[1686,227]
ARQ-Modulo-4	[148, 349]	[205, 312]	[241, 300]	[383, 298]
ARQ-Modulo-8	[168, 349]	[237, 312]	[276, 300]	[408, 298]
ARQ-Modulo-16	[191, 349]	[277, 312]	[322, 300]	[426, 298]
ARQ-DMR	[207, 349]	[302, 312]	[409, 300]	[517, 298]

Table 2.1: Logic synthesis results from XILINX Virtex 5 in terms of Slice/Clock Freq.(MHz). (N,l) as (order of filter, bit length of input)

2.4.2 Pareto distributions for a FIR filter

The Pareto distributions are built from the RE-Spaces that are obtained from the RE-Criteria. Not only the performances in terms of the fault-tolerant and the hardware efficient for different error correction strategies are able to be obtained, but also other features can be predicted. The features for the FIR filters obtained from the different techniques are pointed out in Fig.2.6. More details are shown in Annex. A.4.

Classically, the use of correction techniques results in the degradation of the hardware efficiency, but yields an increase of the reliability. Moreover, different detection methods tune the features, such like, since the spatial DMR based is less efficient than the other Modulo Check methods, but is more reliable. In addition, the more sophisticated error correcting technique is used, the error-resistant performance is better. For instance, the S-TMR based filters that are carried out by the naive method. ARQ via DMR holds high hardware efficiency with high reliability compared to the S-TMR.

However, it turns out that the BRRNS solution can not even bring any interest of error-correction compared to the unitary error-probability p_e. In practical, the BRRNS requires the processes of digit to residue and residue to digit to carry out the RRNS arithmetic. According to (2.15), the restoration of BRRNS, namely the self-diagnosis decoder, requires a huge stack of hardware complexity. Thus, the error probability of decoder block is prone to surpass the unitary error probability p_e. In this case, a fault-tolerant FIR filter is designed by RRNS arithmetic may not as efficient as a simple FIR.

Actually, two methods achieve high reliable feature design: the add-redundancy based method, such as the error control code and the reuse-function based method, such as the ARQ. The ARQ that is not able to correct some errors during the computation can not ensure the data rate. Nevertheless, it is the most efficient and robust one among the three strategies. With the increase of replay time k, its error-resistance sharply increases.

As a consequence, by the construction of Pareto distributions, the designer is able to explore different strategies that depend on the system constraints, such as, hardware resources, power consumption, latency, data rate, etc.

2.5 Conclusion

In this chapter, a Reliability-Efficiency criteria, called RE-Criteria that is a combination of error probability criteria and hardware efficiency criteria has been proposed. Subsequently, conventional mechanisms have also been revisited for the two dimensional analysis based on our criteria. Finally, some Pareto distributions have been built for the error-correcting techniques based FIR filters. This work enable to illustrate the interest of RE-Criteria. As a result, the ARQ mechanism is able to obtain the most reliable circuit by using less hardware overhead among the other strategies, the TMR and the ECC. With the increase of replay time k, its error-resistant capacity sharply increases.

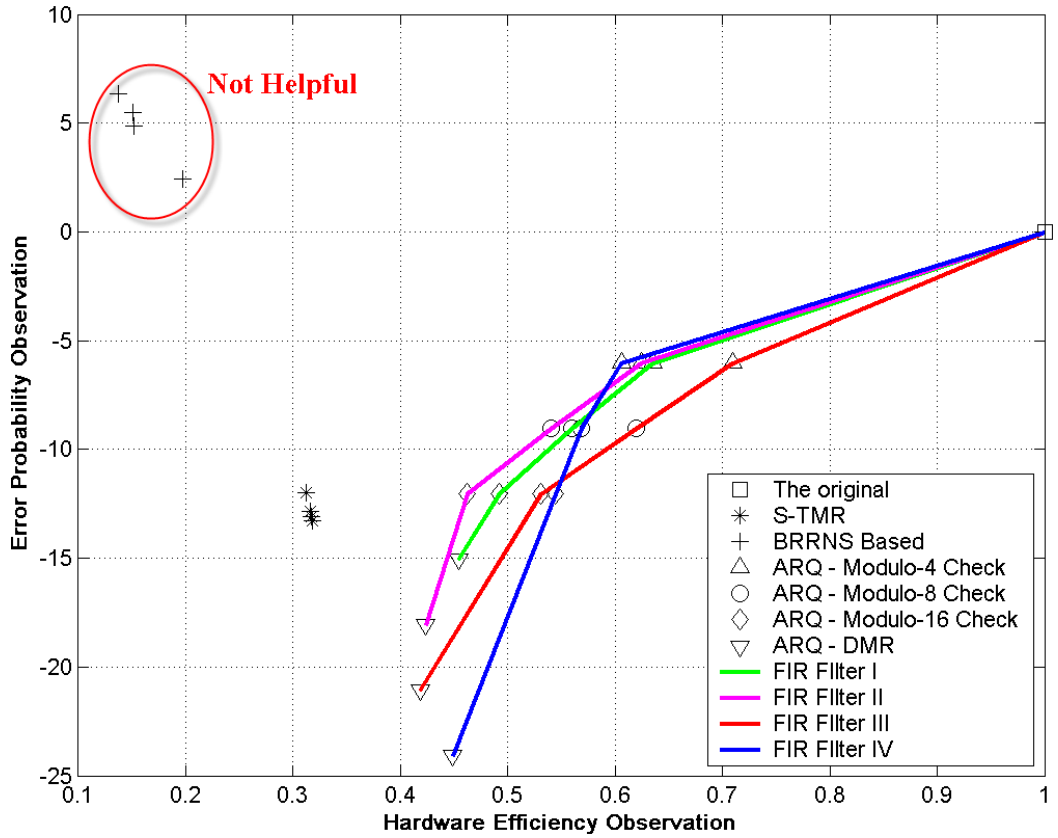


Figure 2.6: Pareto Distributions: the values of error probability and efficiency for different filters are normalized by the ones of the simplex considered as the reference.

Above all, the ARQ requires the system restart the function so as to provide a replayed result to feed into the detector.

Moreover, the BRRNS based fault-tolerant FIR filter can not bring any improvement of robustness, but charge a huge overhead to the design and degrades the reliability as well. It is found that the decoder based on BRRNS requires a huge hardware complexity to carry out the operation in a FIR. In this case, the decoder is more prone to be faulty, and thus, the BRRNS based FIR filter may fail as a candidate for fault-tolerant FIR filter. As such, we lay our emphasis on low-cost and reliability fault-tolerant for embedded system, which are presented in the next chapter.

Since the post/nano CMOS and emerge technology featured electronic devices are concerned as probabilistic circuit, as explained as in previous sections, the design on this kind of circuit can not be considered as reliable any more. For this reason, with using EDA tools to perform a soft-hard co-design (See more details in Annex. A.3), efforts are necessary to pursue a hardware efficient and robust featured design. In general, the conventional circuits are often baffled by unadvisable energy-efficiency/Robustness term. More precisely, it is often difficult to find a efficient way to increase system's relia-

bility. Such like, adding too much extra hardware overhead to the design may make the system clumsy. In fact, an ideal method that achieves the most reliability by the introduction of less hardware overhead as possible, namely energy efficiency. Two curves to depict such tradeoff between energy efficiency to robustness for conventional method and ideal method, respectively, are shown in Fig.2.7.

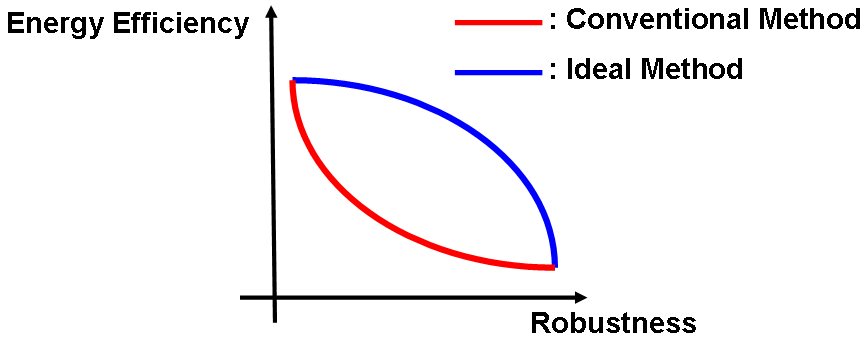


Figure 2.7: Tradeoff between energy efficiency to robustness for conventional method and ideal method.

In order to get low-level analysis results, such like, power, delay, hardware complexity, and etc., the designers have to go through the circuit design process. Thus, the designers have to carry out each circuit design flows independently for different architectures. In this case, enormous time and research resources are wasted. By using this new metric, the designer is able to easily and quickly analyze different strategies and methods, at architectural level, thanks to estimation results in terms of hardware efficiency and robustness.

3

An Embedded Robust Model for Unreliable Binary Circuit

In this chapter, we first introduce some decoding methods for Binary Symmetric Channel (BSC). In the literature, a modified version of the Gallager's well-known Bit-Flipping decoding method has been mentioned for decoding LDPC codes over a BSC in [CV09]. The modification corresponds to the estimation of a variable node to be considered as a majority of the incoming messages after each iteration. In this work, we propose a similar modification, that is to output the majority of the incoming messages only at the final decoding iteration, instead of after each iteration. Using density evolution analysis, the majority-vote modification has a fast convergence. In addition, it achieves significant gain in terms of decoding performance if the variable node degree is equal to three. Since slight extra hardware resource overhead is required, the present method is efficient for a fast low-power error-correction design.

In a second step we review an error-correction model, referred as to Low-Density Parity-Check (LDPC)-coded Fault Compensation Technique (LFCT), proposed by Winstead in [WH09]. The proposed method is designed to reliably compute some operation $F(x)$. Due to the affections by transient/permanent faults, the output of $F(x)$, s, is not reliable. Since an LDPC code is defined by a parity-check matrix H, the valid codeword as a vector $[s\ r]$ can be generated through another encoding function. The sequence r is the redundancy bits of s by the definition of matrix H. Subsequently, a stochastic decoder [GR03, WGRS05] is made to correct errors that occur at the output of some logic computation s. Although a considerable gain in terms of decoding performance at the output is rendered by LFCT, a critical drawback would lead the method to fail. If the correlated errors occur at the output of computing function $F(x)$ and as well encoding function, the stochastic decoding can not perform an efficient error-correction.

To take into account the occurrence of correlated errors, we propose a modified LFCT version, called coded Dual Modular Redundancy (cDMR). Thanks to the crossbar technique, in cDMR the encoding function to generate the redundancy bits r is realized in a correlated-error-free fashion. Moreover, the cDMR is modularized in a formal general robust system.

3.1 Introduction

As digital technologies approach the limits of planar integration, device-level reliability has emerged as a critical concern. Densely integrated CMOS circuits are increasingly affected by thermal upsets which induce momentary transient signal faults in digital architectures. The problem of transient upsets is compounded with the increased use of three-dimensional integrated circuits and the emergence of "post-CMOS" nano-scale devices. To answer to this challenge, it has been proposed to embed error-correcting logic within integrated digital architectures.

At the first place, the error-correction methods for binary system can be considered. In the literature, the well-known Gallager's Bit-Flipping decoding method (GBF) has been introduced by Gallager in 1963 [Gal63] as a decoding method for LDPC codes over a BSC. A modified version of the GBF has been mentioned for decoding LDPC codes in [CV09]. The modification corresponds to the estimation of a variable node to be considered as a majority of the incoming messages after each iteration. In this work, we propose a similar modification, that is to output the majority of the incoming messages only at the final decoding iteration.

Moreover, for the general-purpose robust technique, Winstead proposed the LFCT model in [WH09]. In the presence of transient and permanent defects, the LFCT method is based on Gaudet and Rapley's theory of stochastic decoding [GR03,WGRS05] to correct errors that occur at the output of some logic computation. In [TWB+12, TSW+12], based on the LFCT, we characterize the fault compensation by embedding LDPC codes as a coded dual-modular redundancy (cDMR) technique. The cDMR method is permissible if certain constraints are imposed during the logic synthesis. The required constraints result sometimes in more complex logic syntheses. That can limit the generality of cDMR solutions. TMR provides a more general solution for the protection of black-box logic modules without any constraints on the logic synthesis process. TMR-style solutions have a fixed redundancy of two, whereas the cost of cDMR varies for each synthesized architecture. As such, the characteristics of TMR-style, the overhead of delay and hardware complexity, are predicable. Hence it may be preferable to apply TMR, because the TMR-style provides more determinate characteristics during architecture planning.

The rest of this chapter is organized as follows: we first review some related binary error-correction methods; then a modified Gallager bit flipping method that achieves fast error-correction over binary symmetric channel is introduced; we also review the robust system LFCT. At last, the dual modular redundancy based general-purpose technique cDMR, based on the LFCT, is presented.

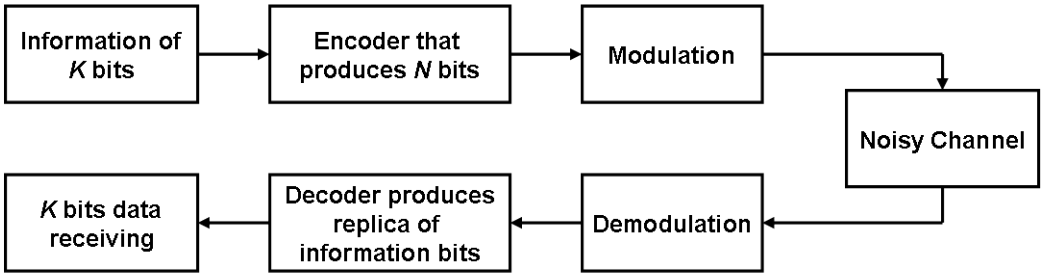


Figure 3.1: Block diagram of a communication system

3.2 Decoding Methods Dedicated to Binary Symmetric Channel

Before the presentation of all the contributions proposed for the embedded binary error-correction, a literature review about the conventional decoding methods is given.

3.2.1 Digital Communication System

The need of efficient and reliable digital data communication systems has been rising during the last decade. The definition of data systems through conventional modulation and voice transmission techniques have generally resulted in systems with relatively low rates and high error probabilities.

According to the Noisy Channel Coding theorem proposed by *C. E. Shannon* in 1948 [Sha48], a digital communication system can be simplified as Fig. 3.1. Shannon introduced the channel capacity $\mathcal{C}$ as a measure of the maximum information that can be transmitted over a noisy channel. More precisely, the main theorem is described as follow: the information is encoded with code rate R. if R is less than the channel capacity $\mathcal{C}$, then it is possible in theoretical to achieve error-free transmission over a noisy channel by an appropriate encoding [Sha48]. By using the set of random codes, it is proved that the existence of a code enabling information to be transmitted with a given error probability. However, it did not give an approach to find such code to achieve the channel capacity.

The information block produces information by the mean of sequences which are row vectors of length K. Then the information data is encoded through an encoder providing a codeword of length N. The code rate R is defined by the ration $R = K/N$. The codeword is after processed by a modulation block to be transmitted over a specified channel with additive noise. After the transmission over a noisy channel, the demodulation process is needed to get received noisy codeword. The distorted codeword is then fed into the decoder. The decoder uses the properties of the code to try to restore the information from the noisy transmission by finding the most probable transmitted codeword, or equivalently the most probable information data.

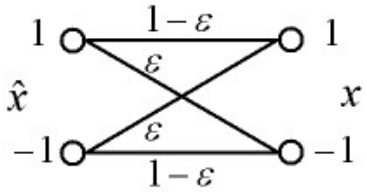


Figure 3.2: Block diagram of a noisy channel BSC.

The source produces binary digits, or bits, of a length K at some fixed time rate. The encoder perform the data processing, modulation, and anything else that might be necessary to prepare the data for transmission over the channel, channel encoding, which is different to the data encoding. The source sequence is separated into blocks of N bits, only one block at once. The data is said to be encoded with rate $R = K/N$. The channel received the data transmitted after modulation, and then disturbs it by random noise. The decoder processes the output of demodulation and produces a delayed estimation of the source bits.

3.2.2 Binary Symmetric Channel

The Binary Symmetric Channel (BSC) (Fig. 3.2) is a memoryless channel model with a binary symbol alphabet $\lambda = \{\pm 1\}$. Let $\hat{x} \in \lambda^N$ and $x \in \lambda^N$ be the modulated channel input and the detected channel output sequences, respectively. The BSC channel is characterized by a cross-over probability, $\varepsilon = \Pr(x \neq \hat{x})$. If $x \neq \hat{x}$, an error is occurred, hence ε is the channel's uncoded Bit Error Rate (BER). To reduce the BER, a data sequence u of length K is encoded before the transmission to yield the codeword c of length N. Thus, this codeword is modulated to produce $\hat{x}$. At the receiver side, a decoder processes the transmitted data x and produces corrected bit z.

3.2.3 Low Density Parity Check codes

Low-density parity-check (LDPC) codes are a class of linear block codes, first introduced by *Robert G. Gallager* in 1960 [Gal63]. Their main advantage is that they provide a performance which is very close to the channel capacity. Classically, we can define it via their parity-check matrices and generator matrices or *Tanner graph* representation.

As other linear block codes, LDPC codes are defined by a Parity-Matrix $H(n, j, k)$ and a Generator Matrix G [Gal63]. The parity-matrix H represents a code of block length N with a matrix $M \times N$ likes the one of Fig.3.3, where each column contains a small fixed number j of 1's and each row contains a small fixed number k of 1's. Two conditions $j \ll M$ and $k \ll N$ that have to be satisfied. In order to achieve this property, the parity check matrix should usually be very large. So the example of Fig. 3.3 and Fig. 3.4 can not be really called low-density.

Not only described by the parity check matrix, *Tanner Graphs* [KFL01], bipartite

$$\begin{vmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{vmatrix}$$

Figure 3.3: (10,2,4) LDPC code parity-check matrix.

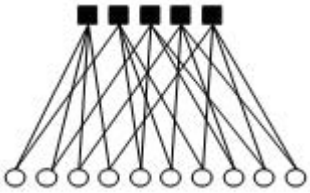


Figure 3.4: (10,2,4) LDPC code bipartite graph.

graphs, introduced an effective graphical representation of LDPC codes. It means that the nodes of the graph are separated into two distinctive sets. Edges connect nodes of two different types. The two types of nodes in a Tanner Graph are called **Variable Nodes** (White Circle) and **Check Nodes** (Black Square). Fig.3.4 is an example of a *Tanner Graph* that corresponds to the code given in Fig. 3.3. It consists of M check nodes (the number of parity bits) and N variable nodes (the number of bits in a codeword).

A Check node is connected to a variable node if the corresponding element of the Parity-Matrix is a 1. If we put the sparse matrix in the form $[P^T I]$ via Gaussian Elimination the generator matrix G can be calculated as $G = [IP]$, where I is identity matrix. Actually, the example given in Fig. 3.3 and Fig. 3.4 is a regular LDPC Codes. Indeed, if j is constant for every column and $k = jM/N$ is also constant for every row. Otherwise, if Parity-Matrix is low density but the numbers of 1's in each row or column are not equivalent, the code is called irregular LDPC Code.

3.2.4 Message-Passing Decoding

Message-Passing Decoding (MPD) method has been characterized in the work of Richardson and Urbanke [RU01]. It is traditionally illustrated as Message-Passing on a code's *Tanner graph*.

Definition 4. *MPD's behavior can be regarded as: Messages care exchanged between nodes in the graph along the edges in the graph in discrete time steps.*

A *Tanner graph* contains two sets of nodes – variable nodes v_i, $i = 1, \ldots, N$, and check nodes p_j, $j = 1, \ldots, M$, where $M = RN$. Edges connect the variable nodes to the check nodes, and as well the check nodes to the variable nodes. The pattern of edge connections is specified by the code's parity-check matrix H. There is an edge between

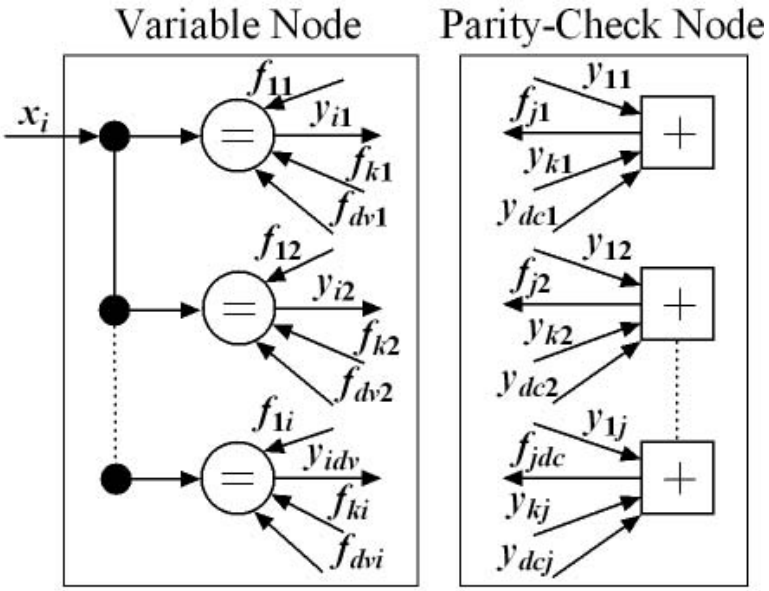


Figure 3.5: Structure of the *variable* and *parity-check* nodes that are contained in an message-passing decoder. The extrinsic information principle is applied, so that an output for a given edge j is computed from all inputs except j.

v_i and p_j if and only if $h_{i,j} = 1$. The degree of a node is the number of edges connected to it in the corresponding Tanner graph.

Let d_v be the degree of a variable node v_i. Let $\mathcal{V}_i$ be the set of edges that are connected to v_i. Similarly, let d_c be the degree of check node p_j. Let $\mathcal{P}_j$ be the set of edges that are connected to p_j. During the decoding process, binary messages are exchanged between the two sets of nodes. The message exchanged from node v_i to p_j is written $y_{ij} \in \lambda$, and the corresponding message from p_j to v_i is written $f_{ji} \in \lambda$, as shown in Fig. 3.5. All nodes are processing functions that follow the standard extrinsic information principle. The outgoing message for an edge k is computed using information from all edges *except* k. This principle ensures that information is not merely echoed back incoming the decoding algorithm, unlikely information is not propagated in the *Tanner graph*.

3.2.5 Gallager Bit-Flipping Decoding

Bit-flip methods were among the first LDPC decoding algorithms introduced by Gallager in [Gal63].

Definition 5. *The Gallager Bit-Flipping Decoding methods (GBF) are defined for frames that were transmitted over a BSC, as a binary hard decision message-passing decoding method [RU01].*

The GBF algorithm can be described as follows. The variable nodes initially transmit messages $y_{ij} = x_i$ for all i, j values. Then, for each check node, the outgoing message for edge k_{edge} is equal to the modulo-2 sum of all local incoming messages (*excluding* edge k_{edge}). A sum is computed separately for all edges associated with all check nodes. Then, for each variable node, the outgoing message along edge k_{edge} is equal to x_i unless at least b_l incoming check messages (excluding edge k_{edge}) disagree with the x_i. There are two well-known cases of the GBF algorithm. If $b_l = (d_v - 1)$, i.e. if all local incoming messages are unanimous, then the GBF corresponds to the traditional *Gallager-A* algorithm. If b_l is a fixed constant (lower than $d_v - 1$), it corresponds to the *Gallager-B* algorithm [Gal63].

The error correction algorithm can be described as follow:

1. Initialize $y_{ij} = x_i$ (the channel-side message), for all i.
2. Compute $f_{ji} = \oplus_{m \in P_{j \setminus i}} y_{mj}$ for all j.
3. For a given variable node v_i, let $f_0, \cdots, f_{d_v}$ be the locally received messages for this node, where $f_0 = x_i$ (the channel-side message). $f_1, \cdots, f_{d_v}$ are the message from check nodes. The output is $-f_i$ if $|\{g \in \mathcal{V}_i \setminus j : f_{gi} \neq x_i\}| \geq b_l$, f_0, otherwise.
4. Iterate steps 2 and 3 during a fixed number of iterations.
5. For the sake of facility, the corrected output z_i is taken as the output from one of d_v variable nodes.

   Remark 1. *This algorithm is the original GBF method. Actually, the estimate of a variable node is taken as shown in step. 3. In practical, some sophisticate mechanisms can be applied for the decoding decision so as to improve the BER performance. Thus, a fast decoding method for LDPC codes over the BSC, which is presented in next section.*

In order to further investigate the decoding performance for a Gallager-style decoding, a (2048,384) LDPC code is simulated in an Bi-Additive White Gaussian Noise (BiAWGN) [BTMB10] for a binary case. More precisely, the received AWGN channel messages are sampled into a binary set $\{0, 1\}$. Simulated BER performance results are given in Fig. 3.6. Note that the Gallager-A decoder produces no benefits at low SNR. But, it still provides a high decoder performance gain for high SNR.

3.3 Fast Decoding Method over BSC

As introduced in Section 2, the designers try to propose architecture with a good trade-off between the energy efficiency and the robustness. In practice, there are other constraints, such like, high throughput, low latency, and etc. For an embedded error-correction decoder, one of the most important criterion is high speed execution. When

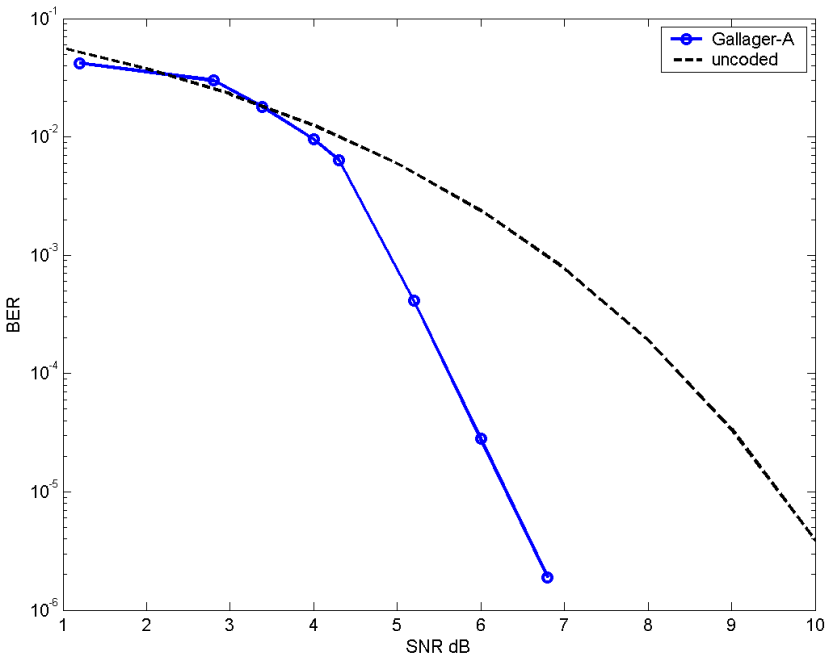


Figure 3.6: Performance of Gallager-A decoder for a (2048,384) LDPC code on Bi-AWGN channel, for 30 decoding iterations.

the performance of error-correction is satisfied, the throughput degradation introduced by the redundancy becomes critical for an embedded system.

In this section, we present a Modified GBF algorithm (MGBF) suitable for all LDPC code structures.

Remark 2. *In [CV09], a modified version of GBF method has been mentioned for decoding LDPC codes over BSC. The modification is that the estimation of a variable node is taken to be the majority of the incoming messages at the end of each iteration. In this work, we propose a similar modification: the majority of the incoming messages is taken at the last decoding iteration as the output.*

We analyze the codes' convergence by the *density-evolution* metric [BRU04]. In addition, we present results for MGBF decoders which consistently achieve better performance than the traditional GBF, especially when the variable node degree is equal to three. To the best of our knowledge, the proposed technique is the most efficient modification for current soft message-passing decoding over BSC. Although the modification is small, it delivers a significant improvement in terms of BER performance. As such, the MGBF is designed for hardware efficient and robust system.

3.3.1 The MGBF Method

LDPC decoding methods are traditionally illustrated as message-passing on a code's Tanner graph [KFL01]. Let us take the same message-passing notations as presented in Section.3.2.4. The modification introduced in this section that we refer as MGBF, is based on a majority voter to process the decoded data z during the final decision step. During the iterations, the MGBF applies the same update rule as the GBF. Majority logic is applied at the final step, when the decoding decisions are rendered. The MGBF error correction algorithm is detailed as follows:

1. Initialize $y_{ij} = x_i$ (the channel-side message), for all i.

2. Compute $f_{ji} = \oplus_{m \in \{\mathcal{P}_j \setminus i\}} y_{mj}$ for each parity-check node p_j and for each $i \in \mathcal{P}_j$. The notation $\{\mathcal{P}_j \setminus i\}$ refers to the set of edges connected to p_j, *excluding* edge i.

3. For each variable node v_i and for each $m \in \mathcal{V}_i \setminus j$, the outgoing message is

$$y_{ij} = \begin{cases} -x_i & \text{if } |\{g \in \mathcal{V}_i \setminus j : f_{gi} \neq x_i\}| \geq b \\ x_i & \text{otherwise.} \end{cases} \tag{3.1}$$

4. Iterate steps 2 and 3 during a fixed number of iterations.

5. The decoded output is $z_i = \text{Maj}\{f_{ij} : j \in \mathcal{V}_i\}$, i.e. the decision is a simple majority of incoming messages. This is similar to step 3 with $b_i = \lfloor d_v/2 \rfloor$. Note that for this step, b_i is not constant if the code has an irregular d_v distribution.

3.3.2 Threshold Determination

In this section, we apply density evolution analysis to determine the *threshold* of MGBF decoders, defined as the maximum $\varepsilon_{\max}$ for which error-free communication can be achieved by the MGBF algorithm applied to a specified degree distribution [RU01].

In this analysis, it is usual to assume that the correct channel inputs are $\hat{x}_i = 1$ for each $i = 1, \ldots, N$. Then during the decoding process, any message equals to -1 is considered as incorrect. Let $Pb_{-1}^{(0)}$ and $Pb_1^{(0)}$ be the probabilities for a variable node during the first iteration that y_{ij} is equal to -1 and $+1$, respectively. By assumption, $Pb_1^{(0)} = 1 - \varepsilon$ and $Pb_{-1}^{(0)} = \varepsilon$. Similarly, let $Pb_1^{(l)}$ and $Pb_{-1}^{(l)}$ denote the probabilities for a variable node, at the l^{th} iteration, that y_{ij} is equal to $+1$ and -1, respectively. Likewise, let $q_1^{(l)}$ and $q_{-1}^{(l)}$ represent the probability for a check node, during the l^{th} iteration, that f_{ji} is equal to $+1$ and -1, respectively.

$P_G^{(l)}$ represents $Pb_{-1}^{(l)}$ in the case of GBF algorithm, that is derived as following:

$$q_1^{(l)} = 1 - q_{-1}^{(l)} = \frac{1 + \left(1 - 2P_G^{(l-1)}\right)^{(d_c - 1)}}{2} \tag{3.2}$$

$$P_M^{(l)} = \varepsilon \cdot \left(1 - \sum_{i=b_g}^{d_v} \binom{d_v}{i} \cdot \left(q_1^{(l-1)}\right)^i \cdot \left(q_{-1}^{(l-1)}\right)^{d_v - i} \right) + (1-\varepsilon) \cdot \sum_{i=b_g}^{d_v} \binom{d_v}{i} \cdot \left(q_{-1}^{(l-1)}\right)^i \cdot \left(q_1^{(l-1)}\right)^{d_v - i} \tag{3.3}$$

From 3.3, if ε is less than a threshold value, the resulting error probability can be approximated to 0, when the number of iteration goes towards infinity, i.e., $\lim_{l\to\infty} P_G^{(l)} = 0$. The parameters as the threshold values were given in [Gal63] and [RU01].

Note that for the MGBF approach, the majority set applied at the end of the decoding process can be considered as an additional redundancy for the decoding decision. Therefore, let $P_M^{(l)}$ denotes $Pb_{-1}^{(l)}$ in the case of MGBF algorithm. This probability only differs during the last iteration. More precisely, $q_1^{(l)}$ and $q_{-1}^{(l)}$ keep the same equations as (3.2), but

$$P_M^{(l)} = \varepsilon \cdot \left(1 - \sum_{i=b_g}^{d_v} \binom{d_v}{i} \cdot \left(q_1^{(l-1)}\right)^i \cdot \left(q_{-1}^{(l-1)}\right)^{d_v - i} \right) + (1-\varepsilon) \cdot \sum_{i=b_g}^{d_v} \binom{d_v}{i} \cdot \left(q_{-1}^{(l-1)}\right)^i \cdot \left(q_1^{(l-1)}\right)^{d_v - i} \tag{3.4}$$

3.3.3 Convergence Analysis

To take into account of the potential performance gain and as well the fast decoding achieved by MGBF, we propose a convergence analysis by considering the evolution of error probability that depends on the number of decoding iteration. In (3.3) and (3.4), if the channel parameter is fixed the output error probability can be determined in function of the number of iteration. As the plane of the error-probability to the number of iteration as shown in Fig. 3.7, for code rates (5,10), (3,6), (4,8), (3,5), (4,6), and (3,4), the channel parameters ε are set up as 0.024, 0.024, 0.030, 0.036, 0.036, and 0.060, respectively. They are close to half of their limits by GBF decoding. From Fig. 3.7, it is shown that by comparison with GBF, the proposed method achieves fast decoding since the curves converge faster at low error probability (approximating to 0) for a same number iterations. In addition, a significant gain is achieved in the case of $d_v = 3$.

3.3.4 Simulation Results

Since convergence analysis shows code's performance in infinite size, simulation results for finite size codes are experimented to demonstrate the decoding performance gain obtained by MGBF. LDPC codes of $d_v = 3$ are considered during three decoding iterations over a BSC with various parameter ε as shown in Fig. 3.8. Obviously, the improvement yield by MGBF decreases while the number of decoding iteration is few. While fast decoding process is applied, the gain with few decoding iterations makes this

variant *Gallager Bit-Flipping* method suitable for implementation with little hardware extra overhead.

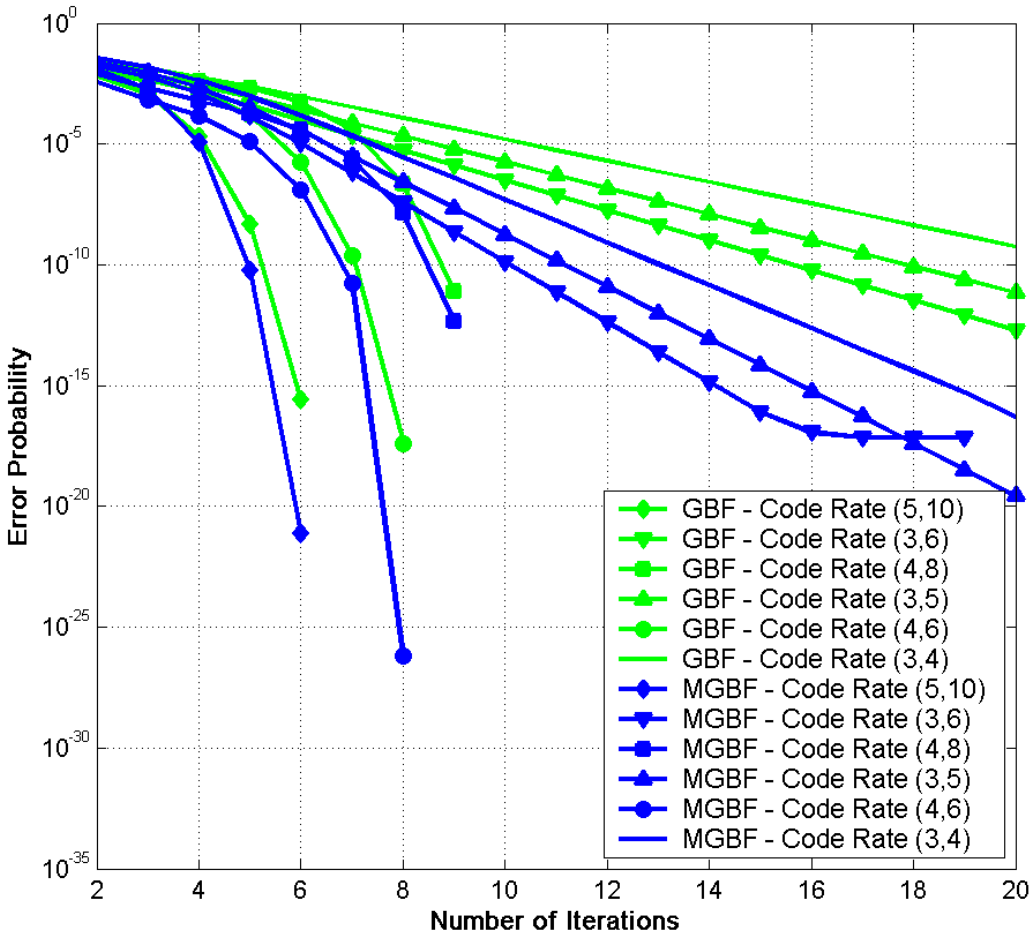


Figure 3.7: Convergence analysis by the evolution of error probability as a function of the number of decoding iteration, under an error-free decoding process.

3.3.5 Conclusion

By using an additional majority voter at the end of the decoder's output, the Modified *Gallager's Bit-Flipping* method (MGBF) is proposed in this subsection. The gain in terms of BER performance achieved with the MGBF by comparison with original *Gallager's Bit-Flipping* method are presented as twofold. Over BSC, MGBF largely increases the convergence time, namely the fast decoding. In addition, a significant gain is achieved in the case of $d_v = 3$. MGBF yields decoding performance gain as multiple factor if the number of decoding iterations is few. Since this significant improvement is achieved thanks to a fast decoding process, this method is attractive for embedded

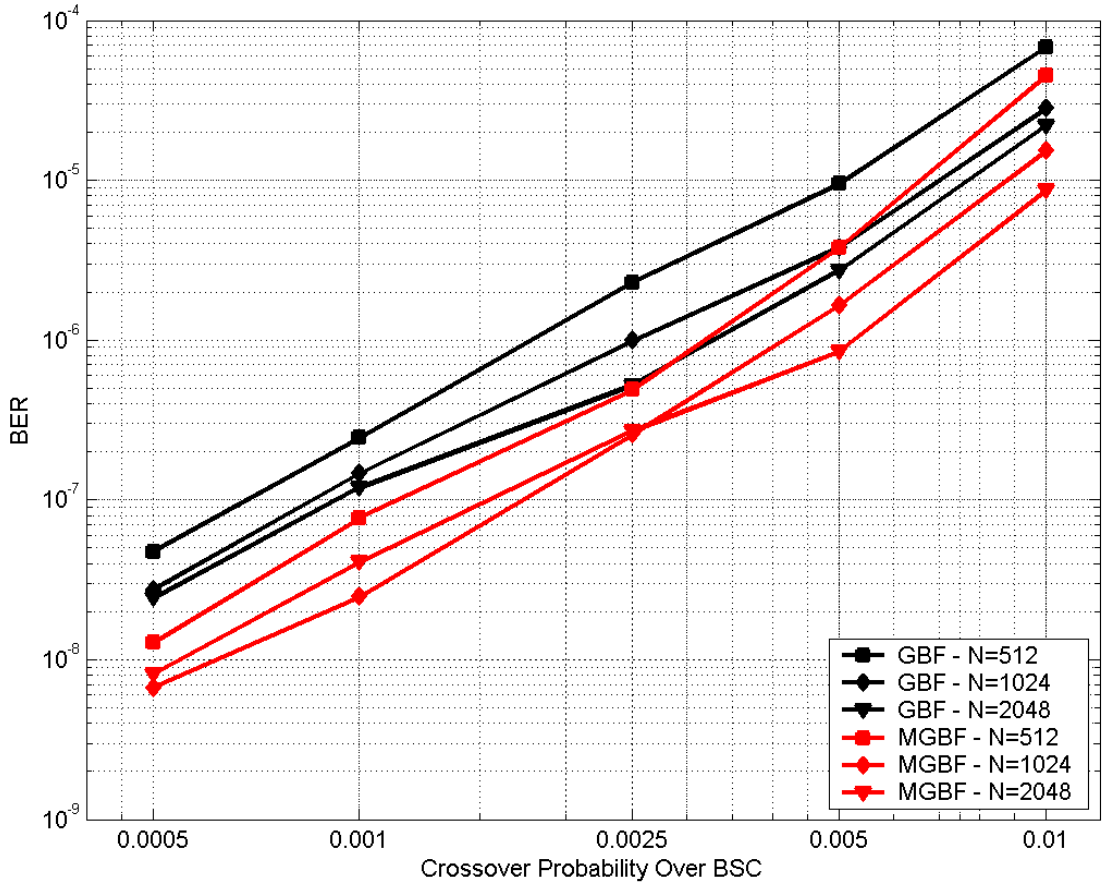


Figure 3.8: BER performance for (3,6) LDPC codes, of which codeword length N are 512, 1024 and 2048. Various parameter ε during three iterations are considered under an error-free decoding process.

error-correction design.

3.4 A Embedded Error-Correction Model

3.4.1 LDPC-coded Fault Compensation Technique

Winstead proposed a LDPC-coded Fault Compensation Technique (LFCT) in [WH09], as illustrated in Fig. 3.9. This technique is relevant to higher LDPC codes resulting in a more powerful error-correcting capacity. In the presence of transient and permanent defects, the LFCT method is based on Gaudet and Rapley's theory of stochastic decoding [GR03, WGRS05] to correct errors that occur at the output of some logic computation.

In the proposed LFCT system, a logic function $F(x)$ is sensitive to transient internal

upsets and as well permanent defects, that produces one or more errors. To correct these errors, a second function $E \cdot F_2(x)$ is introduced which maps the input data s (the output of $F(x)$) to a vector of parity-check bits r, so that the output $[s\ r]$ forms a codeword according to a traditional block error correction design. More precisely, since an LDPC code is defined by a parity-check matrix H, the valid codeword as a vector $[s\ r]$ can be generated through another encoding function. The sequence r is the redundancy bits of s by the definition of matrix H. $F_2(x)$ is the copy of functions $F(x)$. Moreover, the permutation Π is defined by parity-check matrix H, which specifies the set of variables involved in each parity-check equation. Hence, the encoding operation to be the explicit instances F, Π, and XOR logic operations as shown as $\oplus$ in Fig. 3.9, is considered.

Subsequently, a stochastic decoder [GR03, WGRS05] is made to correct errors that occur at the output of some logic computation s. The stochastic decoding method is achieved by equalization modules, as represented by $=$ in Fig. 3.9. The operation for equalization module process is detailed in [GR03, WGRS05]. Thanks to an interleaver Π^{-1}, the stochastic decoder's interconnect corresponds to the code's *Tanner Graph*, which is synthesized using well-known rules [KFL01].

In this system, we assume that the input vector s is error-free. If the number of errors in $[s\ r]$ is sufficiently small, then they are corrected by the decoding process. It results in a correct output vector $[\hat{s}\ \hat{r}]$. Finally the corrected data z can be propagated to other logic modules for additional computations. Although a considerable gain in terms of decoding performance at the output is rendered by LFCT, a critical drawback would lead the method to fail. If the correlated errors occur at the output of computing function $F(x)$ and as well encoding function $E \cdot F_2(x)$, the stochastic decoding can not perform an efficient error-correction.

3.4.2 Coded Dual-Modular Redundancy

By exploiting the principle of LFCT, a coded Dual-Modular Redundancy (cDMR) technique [TWB+12, WTB+12, TBW+13] can be defined as shown in Fig. 3.10. The MPD as introduced in Section. 3.2.4 is taking placed to correct the errors from the output of $F(u)$.

A logic function $F(x)$ is implemented using a digital technology that is subject to errors at its output. The original function $F(x)$ is augmented by the addition of a redundant parity-generator module, $E \cdot F(x)$. E represents the encoding function that generates parity bits codeword space at the output of $F(x)$ as in [WH09]. The *systematic* output word s with a length K from $F(x)$ is then concatenated with the parity outputs r from $E \cdot F(x)$, yielding a complete codeword $[s\ r]$ of length N. If there are no errors, then the resulting codeword should satisfy the standard parity-check constraint, $[s\ r]\, H = 0$, where H is the parity-check matrix.

According to the code's H matrix, the MPD is synthesized by parity-check-node modules and variable-node modules, that are indicated in Fig. 3.10 by $+$ and $=$, re-

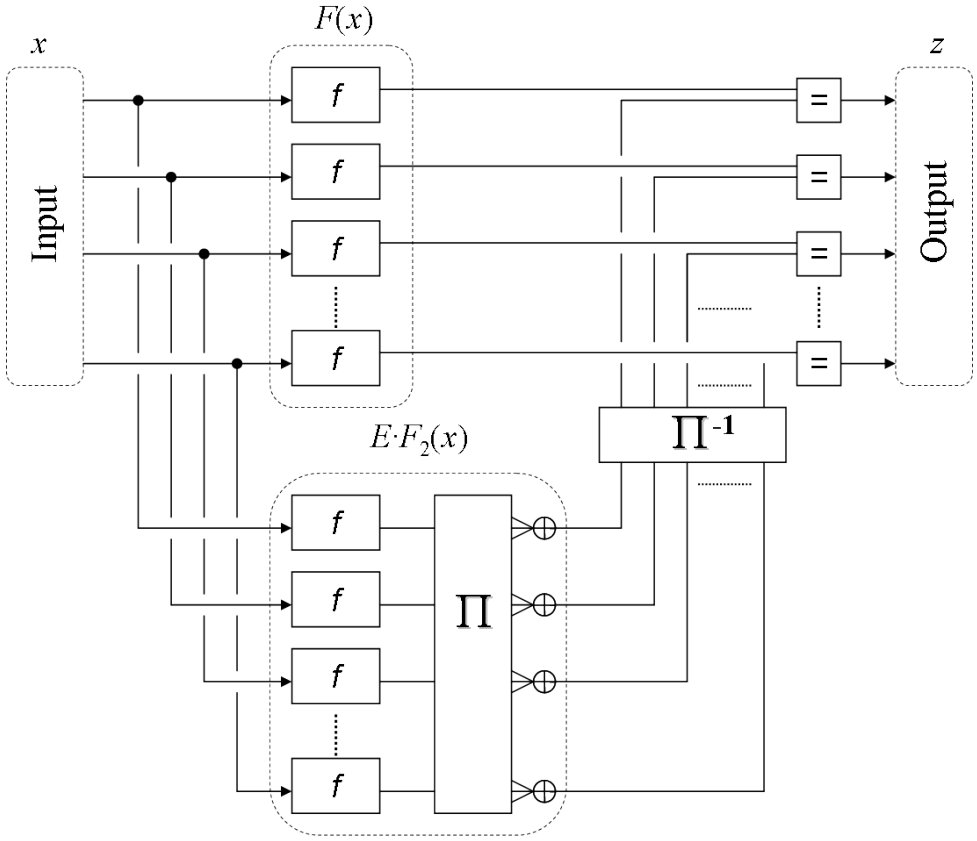


Figure 3.9: Architecture of the LFCT method. The original logic function $F(u)$ is associated with a block ecoding function. Outputs are then corrected by the stochastic decoder.

spectively. Based on the code's Tanner Graph, the interconnects correspond to an interleaver, that is synthesized from well-known rules [KFL01]. The system's final decisions z are taken from the MPD.

In order for the MPD method to function properly, the function $F(x)$ and the encoding function logic $E \cdot F(x)$ must be presented into the duplicate functions. Thus, correlated errors among the information bits and the parity bits are avoided. If the architecture were implemented, then a SET in $F\,(x)$ or in $E \cdot F(x)$ would generate several errors in the informations bit or/and the parity bits, namely correlated errors in output. The function $F(x)$ and the composite function $E \cdot F\,(x)$ have to be synthesized as a flat Boolean function to avoid correlated error patterns. The most reliable approach is to use a flat truth-table synthesis, as is done with cross-bar logic arrays [ROK09]. This approach guarantees that error events occur independently on all the codeword bits, so that correlated error bursts are precluded.

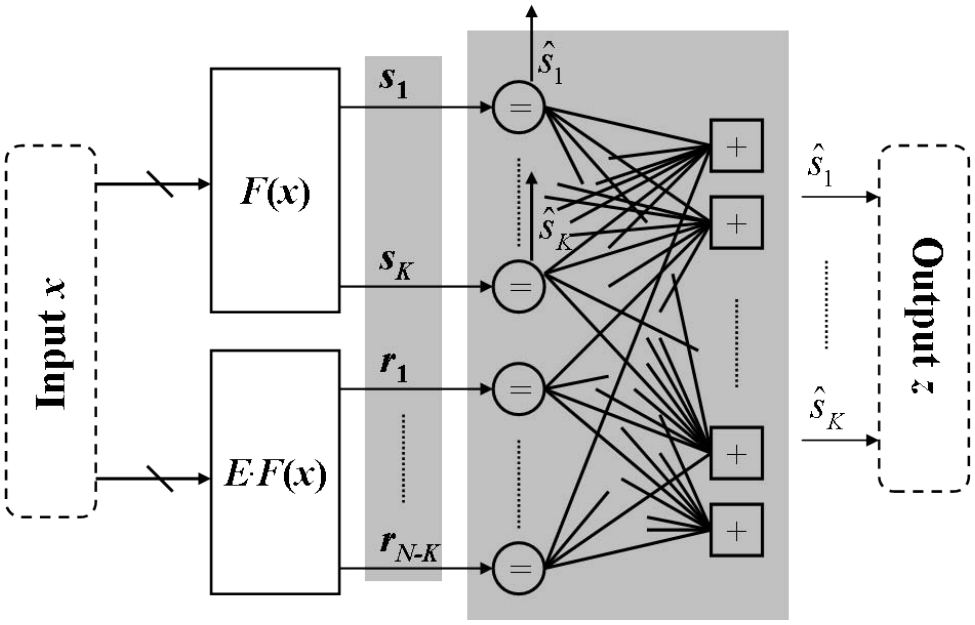


Figure 3.10: The architecture of cDMR.

3.4.3 The Implementation for cDMR

To evaluate the redundancy cost of the proposed architecture, we suppose that $F(x)$ is a large block of crossbar logic (i.e. gate array logic). It represents a minimized sum-of-products expression. The crossbar fabric may consist, for example, of a layer of AND operations followed by a layer of OR operations. It may be used for flat truth-table synthesis. This style of logic is generally not optimal in terms of logic gate number. But, it is useful in our analysis for three reasons.

First, crossbar logic is quite popular in research on next-generation, post-CMOS logic implementation, and reliability is a key concern for crossbar implementations [ROK09]. In the crossbar implementation, as shown in Fig. 3.11 [TBW+13], logic gate is implemented by associations of AND-logic gates and OR-logic gates. Fig. 3.11-a represents a traditional ripple-carry adder where the stars indicate the occurrence of an error that is propagated to multiple signals. For some emerging electronic device, the ripple-carry adder can be designed as shown in Fig. 3.11-b. The "dots" in Fig. 3.11-b indicate the placement of junctions which physically implement the logic operations. In this type of implementation, each operation is associated with a single output. If a momentary fault occurs at some junction, it will propagate only to a single output. This type of logic guarantees that single-error events are correctable.

The major disadvantage of crossbar logic is that the operation counts are not optimal. The crossbar adder in Fig. 3.11-b, for instance, has 57 separate operations. Crossbar logic does not generally obtain minimized gate complexity, but it enables an improvement of reliability by eliminating error propagations. The gate complexity of

the adder could be considered as a form of redundancy. Furthermore, if crossbar-style logic is used for the cDMR method, then the logic function can be duplicated only once to provide error correction.

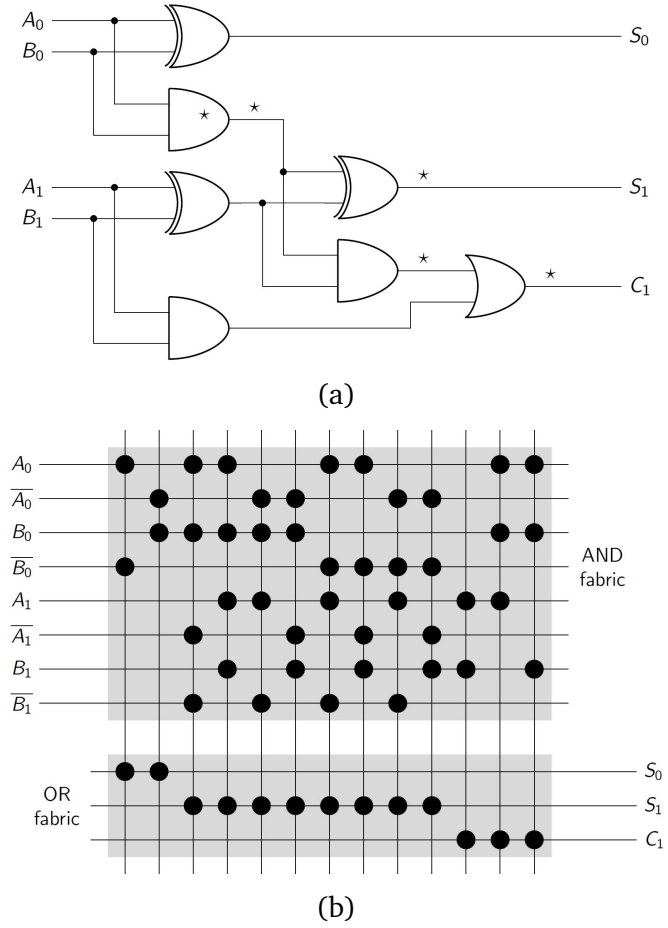


Figure 3.11: Implementations of a two-bit binary adder function based on a traditional ripple-carry design (a), and a crossbar design suitable for some nanoelectronic device families (b). The '$\star$' symbol indicates the occurrence of an error that is propagates to multiple signals.

The second reason is that a single gate-level fault will typically propagate to a single output of $F(x)$. In alternative combinational styles, a single gate fault may be propagated catastrophically to many outputs. The MPD architecture is likely to fail in such cases of sudden error bursts. Therefore, we have to constrain its application to logic styles such as crossbar fabrics where such bursts are precluded.

The third motivation for considering crossbar logic is that it provides a precise strategy for the synthesis of the auxiliary function $E \cdot F(x)$. For arbitrary functions $F(x)$, the average complexity in terms of logic gates to synthesize $E \cdot F(x)$ is close to the aver-

age number of logic gates necessary to synthesize $F(x)$. Hence the method requires an average extra complexity of approximately two, plus an additional cost to implement the decoder.

3.5 Conclusion

In this chapter, the binary error-correction issue is addressed. We first review some related binary error-correction methods. The message-passing decoding method is known as the decoding method over a BSC. One of the message-passing decoding family, Gallager Bit-Flipping method (GBF) proposed by Gallager for LDPC decoding purpose [Gal63], is also recalled.

After that, we present two solutions for embedded binary error-correction on unreliable circuit:

- **A Fast Decoding Method for Binary Symmetric Channel.** A mechanism to refine decision for decoding LDPC codes over a BSC has been studied. The proposed algorithm is a modified version of GBF method. The modification corresponds to the estimation of a variable node to be considered as a majority of the incoming messages at the final decoding iteration. Our modification is similar to the one made in [CV09], a majority of the incoming messages after each iteration. Using density evolution analysis, the majority-vote modification by us has a fast convergence. In addition, it achieves significant gain in terms of decoding performance if the variable node degree is equal to three. Since slight extra hardware resource overhead is required, the presented method is efficient for a fast low-power error-correction design.

- **A General Embedded Error-Correction Model.** The coded dual-modular redundancy is based on the DMR. but a parity-mapped function is used instead of a duplicated original function. This robust model is a modified version of the technique proposed by Winstead in [WH09], LDPC-coded Fault Compensation Technique (LFCT). Suppose that computing functions are vulnerable to transient faults and permanent defects. In the LFCT system, it is desired to reliably compute some operation $F(x)$ with the output s. To achieve this end, according to the parity-check matrix H an extra encoding function $E \cdot F_2(x)$ is designed to produce the redundancy bits r for the information bits s. $F_2(x)$ is the copy of function $F(x)$. Such that, a stochastic decoder is then employed to correct errors from the codeword, a concatenation of $[s\ r]$. However, a critical drawback would lead the LFCT method to fail. If the correlated errors occur at the output of computing function $F(x)$ and as well encoding function $E \cdot F_2(x)$, the stochastic decoding can not perform an efficient error-correction.

  In the modified version of LFCT, cDMR, both computing function $F(x)$ and parity-mapped function $E \cdot F(x)$ are realized in the fashion where the correlated errors

are precluded. Moreover, the cDMR is modularized in a formal general robust system.

However, for the ECC block of cDMR, the occurrence of internal transients within decoder itself have not been considered yet. Obviously, the resulting error probability is influenced by a noisy decoding process. Neither the GBF method nor the stochastic method can perform an efficient error-correction under a faulty decoding process. To take into account the reliability issue of a decoder, the decoder against internal transient faults is presented in next section.

4

A Decoder Reliable Against Internal Transient Faults

This embedded error-correction model, cDMR, can be adapted into any unreliable design whenever two issues are avoided: correlated errors in original function's output and also the parity-mapped function's output, and the ECC circuit is able to perform efficient error-correction in the presence of internal transient faults. Thanks to the cross-bar technique, the correlated-error issue can be avoided in cDMR.

In this chapter, we focus on the efficient error-correction in the presence of internal transient faults within decoder itself. We first review two Muller C-element based fault-tolerant methods for robust latch and general-purpose robust model, respectively. Then, two major contributions to the subject of embedded reliable decoding technique in a faulty process are presented as follow.

- **A Decoder Reliable Against Internal Transient Faults, referred as to MCD**
  A decoding algorithm and logic implementation are proposed for low-complexity error correction in environments with a high rate of transient faults as well as hard errors. The decoder architecture is able to correct a single error in one clock cycle. This feature makes it suitable for mitigating faults in pipelined digital logic architecture. The proposed method is also resilient against internal transient gate errors that may occur within the decoder itself. In fact, thanks to the C-elements, this decoder is robust against internal transient faults.

- **A Space-Time Redundancy Technique to Improve the Decoding Performance**
  A temporal majority logic is applied at the decoder's output, thanks to additional dimension of redundancy. In fact, when applying this space-time redundancy technique, the decoder is able to ground its output error-probability below its internal transient error rate. Moreover, the majority unit is required to be reliable. Finally, we propose a structure to design a low cost reliable majority voter. As a result, by applying space-time technique, it significantly increases embedded decoder's BER performance even under a faulty process.

4.1 Introduction

Embedded error correction may be used to mask permanent circuit defects as well as transient faults. Unfortunately an embedded error correcting decoder has to be designed using the same error-prone device as the functions to correct. This introduces the problem of decoder architectures that are able to detect and/or correct faults in their own execution. There is a list of publications about this topic, with solutions in the form of "self-checking checkers." Self-checking logic circuits are traditionally designed using formal logic methods, where it can be proved that the circuit detects or corrects up to a fixed number of faults.

In the literature, some researchers also studied the resilience of iterative decoding algorithms implemented with faulty internal logic [VC07,LPG12]. In this work, we propose an alternative analysis approach based on *Probability Signal Flow (PSF)*. The PSF analysis was previously proposed as a method to embed LDPC codes [Gal63] for fault compensation in nano-scale digital logic [WH09]. The PSF approach was also applied at the circuit level, resulting in a robust error correction method known as Restorative Feedback (RFB) [WLMT11]. The RFB method is logically identical to the traditional method of TMR, as explained in Section. 1.1.1. However it was shown via a PSF analysis that the RFB method achieves a much lower error rate than TMR in the presence of internal transient upsets. On the contrary, an LDPC stochastic decoding technique has been introduced to cope with internal high-rate fault by requiring redundancy of one under certain constraints [TWB+12]. This technique has been considered as the decoder against internal faults.

The rest of this chapter is organized as follows: we first review some related fault-tolerant methods; to the best of our knowledge, the first decoder against internal transient faults thanks to the Muller C-elements, called MCD, is detailed. then C-element's inherited fault-tolerance is investigated; In order to further reveal the interests of MCD, we study on its threshold determinations under an approximation in the cases of error-free decoding and noisy decoding process; Moreover, a space-time redundancy technique is exhibited for further increasing decoder's performance under noisy decoding process; At last, we present a good decoder candidate for the ECC of cDMR. In addition, the property of dynamic power saving carried by MCD and the analysis via trellis structure is also studied.

4.1.1 Muller C-element and Modified Muller C-element

Muller C-elements [MB59] was introduced for the design of asynchronous circuit. Thus, they were also used for fault-masking in digital circuits [WGRS05,Win09,WEH09, ZMM+06a,TWB+12]. A standard binary C-element circuit is shown in Fig. 4.1. In this

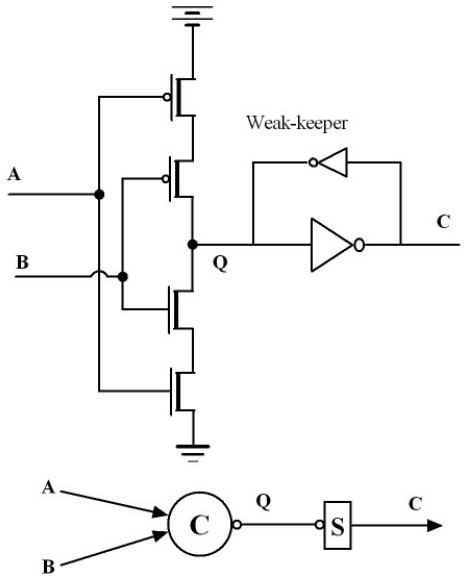


Figure 4.1: A standard Muller C-element circuit, composed of C-not gate and S gate. The C-not gate detects if the inputs are equal. The S gate, the weak-keeper, acts as an inverting state memory.

scheme, the "C-not" gate detects if the inputs are equal, with an output given by

$$Q = \begin{cases} \overline{x}, & \text{if } x == y \\ Z, & \text{otherwise,} \end{cases} \tag{4.1}$$

where Z denotes a high-impedance output state. If $Q \neq Z$, the C-not gate actively drives the S latch by overpowering its state. When $Q = Z$, the state of S is maintained via weak feedback. The further investigation to C-element and its applications are expanded in later parts.

To implement C-element based decoding algorithms or fault-tolerant techniques, it is necessary to use the modified C-element scheme given in Fig. 4.2. This scheme was introduced in [WLMT11] to support error-correcting operations in the RFB method. The modified C-element works in two phases, called "initialization" and "restoration." During the initialization phase, the C-not gate is disconnected so that the S latch can be set to a known initial value. The mechanism to initialize a known initial value via the signal $'z_i'$ is shown in Fig. 4.2. During the subsequent restoration phase, the C-element is activated. The two-phase behavior can be applied to implement Gallager-style error-correction, as shown in the following sections.

Due to its inherent error-resilience ability, C-element is able to prevent the internal faults or the SET. Let us assume that any internal fault induce a glitch to the C-element, such like, when a transient fault occurs one of the inputs, the output of C-element stays stable since the inputs are not unanimous. A simple application for C-element, latch

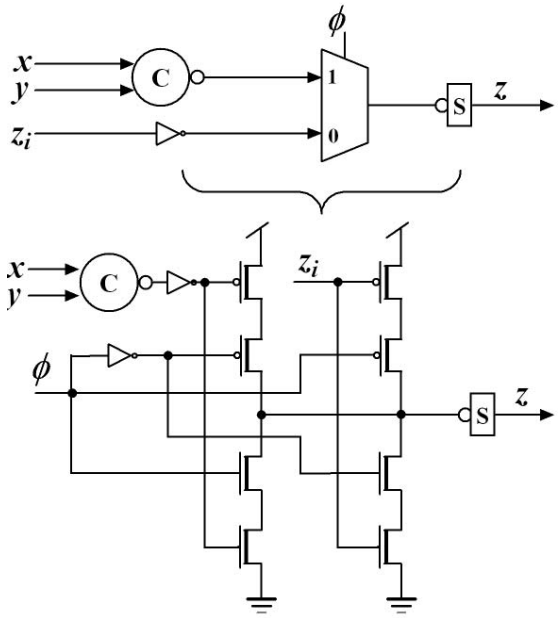


Figure 4.2: A modified Muller C-element based on a two-phase operation. Phase $\phi = 0$ enables initialization of the S latch. Phase $\phi = 1$ corresponds to a normal operation of the C-not gate.

error correction [MZS+07], is given in Fig.4.3. Let two latches store the output from any combinational logic, the C-element that is added after the latches is able to mask any SET from the two latches. The fault-tolerance inherited by C-element is studied in Section. 4.3.3. Sophisticate applications of C-element and its reliability analysis are further studied.

4.1.2 The Restorative FeedBack Method

As presented in Section.1.1.1, TMR methods require a duplication of a logic function twice. Moreover, no constraint for the logic synthesis process has to be given (i.e. TMR methods can correct cases of error-propagation). Since Muller C-element holds an inherent ability of fault-tolerant as shown in Section.4.1.1, a combination of TMR with C-element is proposed in a variant of TMR, referred as Restorative FeedBack (RFB) [WLMT11].

The RFB method is based on Muller C-element gates. The RFB scheme is derived from the principle of stochastic iterative decoding. It enables to achieve better performance than traditional majority-based TMR for correcting temporary upsets. The RFB method is also applicable to M-ary logic architectures as well as conventional binary logic.

An implementation of the RFB method is detailed in Fig.4.4. In this architecture, a multiplexer is inserted between the C-not and S gates. During the setup phase, the

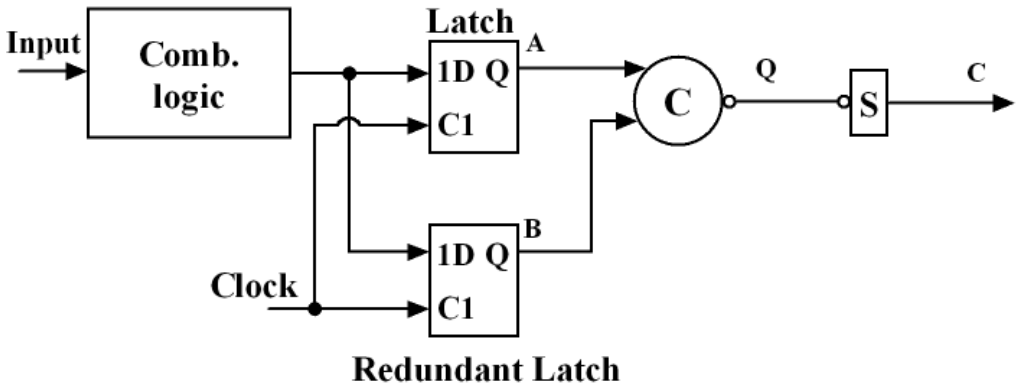


Figure 4.3: Diagram of a latch error correction by using C-element

multiplexer selects the adjacent input signal, so that the state of S is directly forced into an initial value. During the restoration phase, the multiplexer selects the output from the C-not gate, to select the feedback.

A two-phase error correction based on the RFB technique is illustrated in Fig. 4.5. At first, the three repetitive signals, x_1, x_2, and x_3, are fed into three C-elements so as to correct the affected signal x_2. During the initialization phase, the outputs are initialized with barrel-shifted copies of the input signal values. Since x_2 is erroneous, the initialized value of the last C-element is incorrect as well. During the restoration phase, the feedback is activated. After the restoration phase, the output y_3 and the state memory value are both correct.

4.2 Error Model Applied during experimentations

In this section, we illustrate the error model applied in our decoding simulations for the case of noisy decoding process. At first, we recall the error model. An improved error-model is then introduced. At last, we compare the BER performances of GBF decoding method under two different error-injections.

4.2.1 Error Model Used in this work

An error model for the iterative decoding process that has been described in [Var11] is supposed to be the felicitous model to be adapted in a noisy iterative decoding process. In this work, the error model for faulty decoder is considered as error-free message-passing decoder concatenated with an independent and identical BSC. It enables to have a suitable examination of the central phenomenon.

Instead of having a BSC of same parameter ε for each message passing within variable node and check node as assumed in [Var11], two independent BSCs are used in this study. As such, messages produced by the variable node and the check node

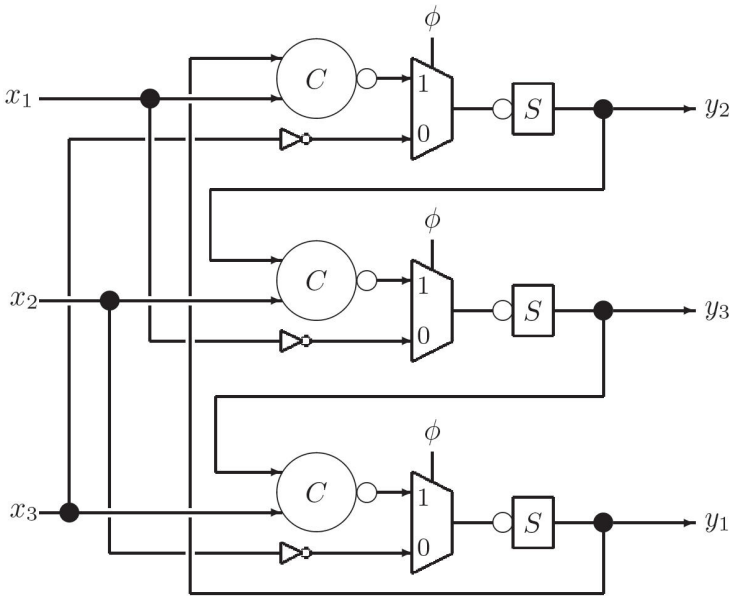


Figure 4.4: Implementation of the restorative feedback method using binary CMOS gates.

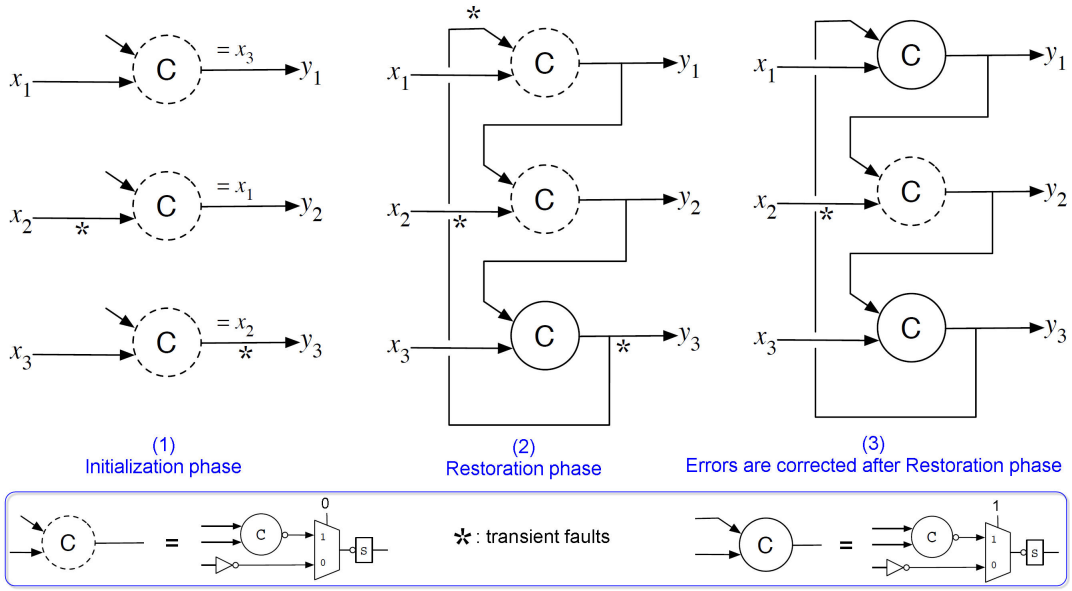


Figure 4.5: Two-phase error correction in the restorative feedback method. The dotted circles indicate C-elements that are inactive. It means that they are not actively driving their outputs. Moreover, the star indicates an error location.

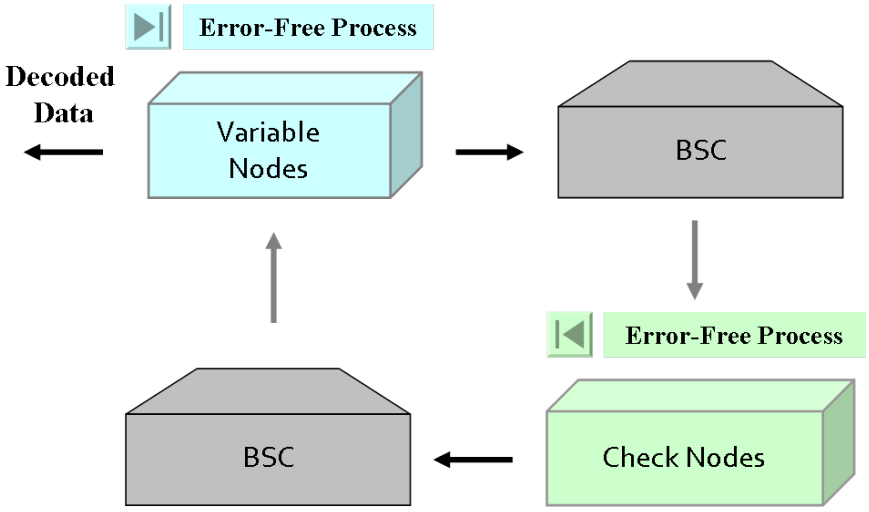


Figure 4.6: An error model for the iterative decoding process.

are exchanged over two BSCs with crossover probability α and β, respectively (Fig. 4.6). Actually, the idea of the error model is basically plotting two independent BSCs between variable node and check node.

4.2.2 Error Injections for Gallager-A Decoding Method

In fact, the noisy decoding model is supposed to be as accurate as possible. Such that, an ideal error model may be detailed down to transistor level. Corresponding to circuit's technology, each transistor is simulated with transient events, intermittent events, and permanent events.

In this work, instead of making a fine model, we simplify the upsets/events and inject transients into key path of the architectures. Since the architectures are close to the implementation circuit, the hypotheses on key path may able to generate most significant error events that would trigger a resulting error. An architecture with error-injections is depicted for Gallager-A decoder, in Fig. 4.7. More precisely, for the architecture of a variable node, two error-injections are involved in the error model. The locations of two error-injections are: received value from channel or the decision of multiplexer to output either the incoming message from check node or received value. For the check node, one error-injection is made at the output of each XOR logic operation.

4.2.3 Comparisons between two different error injection approaches

Note that the error-injection model for Gallager-A in Fig. 4.7 does not exactly match the error model presented in Fig. 4.6. In this subsection, we compare two error-injection models for Gallager-A decoding method. The first one is the error-injection

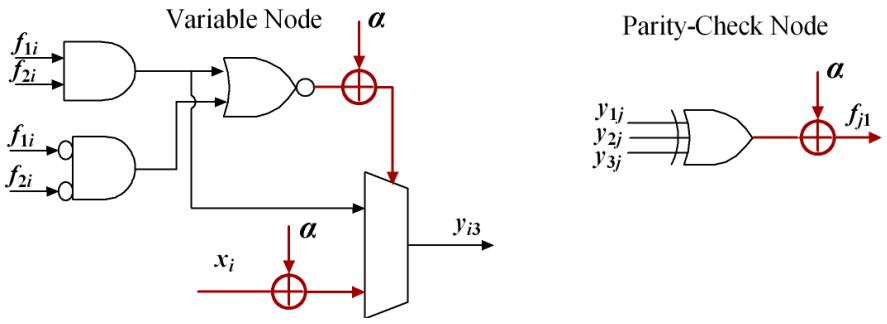


Figure 4.7: Noisy Gallager-A decoder architecture.

model of Fig. 4.7 that are used for all the simulations in this work. The second one is an error-injection model based on the error model of Fig. 4.6.

At first, an error-injection model based on the error model of Fig. 4.6 is shown in Fig. 4.8. As explained in Section. 4.2.1, the error model is inserted in two independent BSCs between variable node and check node. Hence, for the error model in Fig. 4.8, it injects a single error at the output of each variable node operation and check node operation as well.

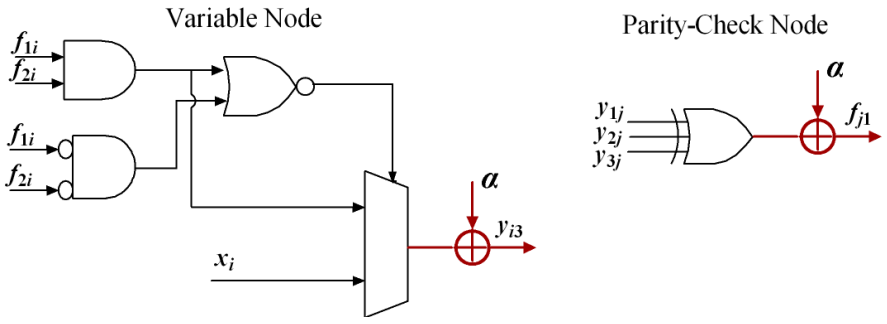


Figure 4.8: Alternative error-injection model for Gallager-A algorithm.

Fig. 4.9 shows the difference between the two error-injection models. From Fig. 4.9, it is found that when the number of decoding iteration is 2, the BER performance difference between two error-injection models are pretty trivial. With an increase of decoding iteration, the difference still stays low. For instance, the maximum gap is observed when $\varepsilon = 10^{-4}$. The two curves still end at the same order of magnitude.

As a result, all the noisy Gallager-A decoding method is simulated with the error-injection model of Fig. 4.7. By comparison to the error-model shown in Fig. 4.6, the error-injection model in Fig. 4.7 does not show a huge gap in terms of BER performance.

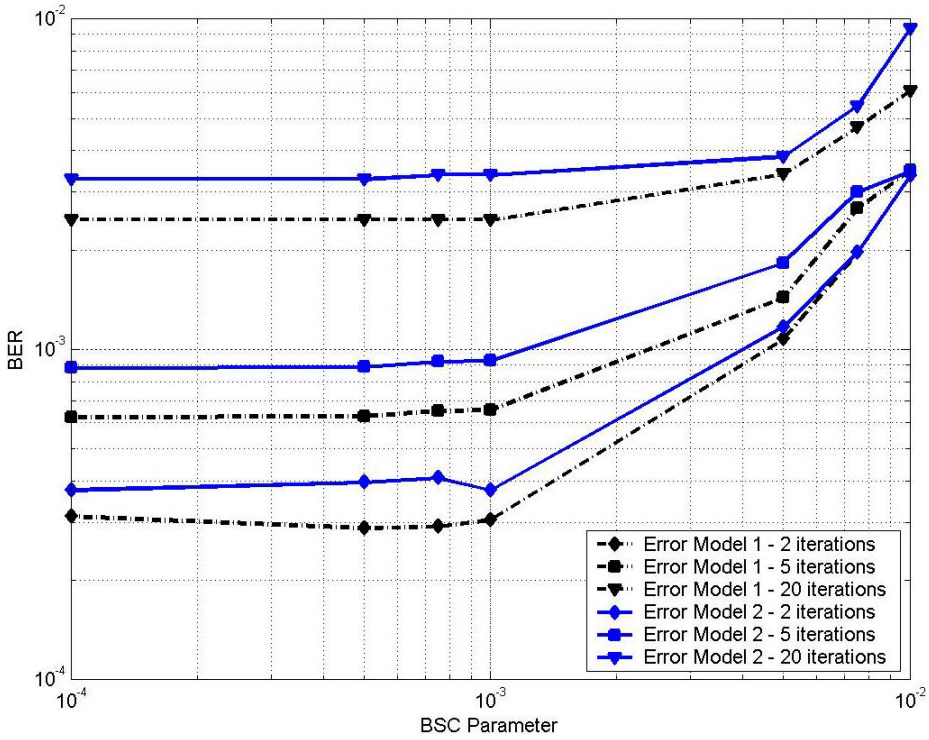


Figure 4.9: Difference of BER performances between two different error-injection models for Gallager-A decoding method of (3,6) LDPC codes of length 64. Model 1 is the error-injection model of Fig. 4.7. Model 2 is the error-injection model of Fig. 4.8.

4.3 MCD: A Decoder Against Internal Transient Faults

In this section, a decoder against internal transient faults is detailed. More precisely, the decoder is able to perform the error-correction under a faulty decoding process, especially in the case of a high error-rate.

4.3.1 Code Structure and Decoding Algorithm

Muller C-element based decoding method, referred as to MCD, has been designed for embedded error-correction in the presence of internal faults [TWB+12, TSW+12, WTB+12]. The proposed decoding method is based on the Tanner graph [KFL01] of a parity-check code, which also can be considered as one of the MPD family. As such, we take the same notation for the Tanner graph as in Section. 3.2.4. In the proposed MCD algorithm, C-element (see presented in Section. 4.1.1) gates are used to implement the variable nodes, as is done with conventional stochastic LDPC decoders [GR03, WGRS05].

Unlike stochastic decoders, our MCD method introduces a new two-phase operation type for the variable node. Fig. 4.10 shows the structure of a variable node, such as, a

cascaded C-element set. In fact, the MCD architecture can be regarded as circuit-level designs for variable nodes in an LDPC decoder, where cascaded C-element sets are used. Please note that the MCD method performs the decoding process in two phases: initialization phase during the first iteration and restoration phase during the rest of the iterative process.

For each variable node v_i, we associate a set of C-element gates C_k, $0 \leq k < (d_v - 1)$, where d_v is the variable degree. Each C-element gate C_k contains a single-bit storage element c_k. The error correction algorithm is described as follows:

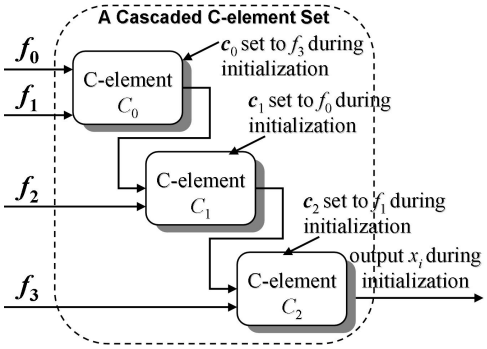
1. Initialize $y_k = x_i$, for all $k \in \mathcal{V}$.
2. Compute $f_{ji} = \oplus_{m \in P_{j \setminus i}} y_{mj}$ for all $j \in \mathcal{P}$.
3. Initialize each C-element memory as $c_k = f_m$, where $m = (k + d_v - 1) \bmod d_v$.
4. The C-element's port connections are as follows. For C_0, the inputs are f_0 ($f_0 = x_i$) and f_1, and the output is c_0. For C_k, the inputs are c_{k-1} and f_{k+1}, and the output is c_k.
5. Iterate steps 2 and 4 during a fixed number of iterations, as the restoration phase.
6. For the sake of facility, the corrected output is taken as $z_i = c_{(d_v-1)}$, the output from one of d_v cascaded C-elements sets.

Each variable node is able to correct each single error during the iteration process. The error location can be in one of its input messages or in one of its internal states. A careful examination of the variable node behavior shows also that it is able to correct several multiple-error patterns. Additional iterations may be used in function of the considered application. The further analysis of error-correction by a single cascaded set is next given.

4.3.2 Error-correction Analysis

An error may originate from an internal error within the decoder. The C-element cascade helps to stop the propagation of such errors. For example, if a transient upset occurs in one of the C-element's state memories, this error is masked by the subsequent C-element in the cascade architecture. In the rest of this subsection we show that single error events are corrected by the MCD algorithm, regardless of where those errors originate.

Two important error cases are examined. The case of single-error event and the case of double-error event. The locations of the errors are examined at the input of variable node or/and the memories of C-elements. In each case, the algorithm is shown to correct the error in a single iteration.



					Initialized			After 1 iteration		
	f_0	f_1	f_2	f_3	c_0	c_1	c_2	c_0	c_1	c_2
Error-free	0	0	0	0	0	0	0	0	0	0
Error in f_0	1	0	0	0	0	1	0	0	0	0
Error in f_1	0	1	0	0	0	0	1	0	0	0
Error in f_2	0	0	1	0	0	0	0	0	0	0
Error in f_3	0	0	0	1	1	0	0	0	0	0

Figure 4.10: Local implementation of a variable node. The architecture is a cascade of C-element gates, modified for the initialization of the state memory. In this figure, $d_v = 4$. Each C-element has three inputs; the left-side inputs are the usual inputs, and the top-side input is the initial state for the C-element's memory. Moreover, a table that illustrates the behaviors of c_k is given.

4.3.2.1 Single-Error Events

A Single Error Appears at x_i If a single error occurs at some input x_i, then the error is initially propagated along several messages y_{ij}. The error is further propagated to the messages f_{jm} for all nodes $m \in P_j$. If the code has no four-cycles (i.e. cycles of length four), no variable node receives more than one erroneous f_{jm} message. Suppose that erroneous messages are sent from variable node i to distinct check nodes j_1 and j_2. j_1 and j_2 both send erroneous messages to the same variable node m. Then, a cycle has to link nodes $i \to j_1 \to m \to j_2 \to i$, which has length four. If four-cycles are excluded, then this cannot occur. Furthermore, no erroneous message is propagated back to variable node i because all other nodes carry correct values. Therefore, each variable node receives at most a single erroneous message among the locally-received messages f_k. As shown in the next subsection, all such single errors are corrected.

Moreover, Fig. 4.11-(a) demonstrates a single cascaded C-element set that corrects the single-error event at x_i. During the initialization phase, the state of C-element C_1 is incorrect due to the initialized value x_i. After one iteration, the error at x_i, as one of the inputs of C_0 is masked by C_0. Additionally, the state of C_1, c_1, is corrected by the two correct inputs.

A Single Error Appears at f_k If variable node v_i receives one erroneous message f_k, then c_m is initialized to an erroneous value, but all other c_k ($k \neq m$) are initialized to correct values. Then, for each C-element, there are three possible cases:

1. The C-element has two correct inputs and an initially erroneous memory state. In this case, the memory state is corrected because the inputs agree.

2. The C-element has a correct memory state but one incorrect input. In this case the correct memory state is retained because the inputs do not agree. Hence, the fault is masked by the C-element's memory.

3. The C-element has two correct inputs and an initially correct memory state. In this case there is no error.

Fig. 4.11-(b) shows the error-correction diagram for single-error event at f_k. As a result, the erroneous f_1, as one of the inputs of C_0 is masked by C_0. Furthermore, the state of C_2, c_2, is corrected by the two correct inputs.

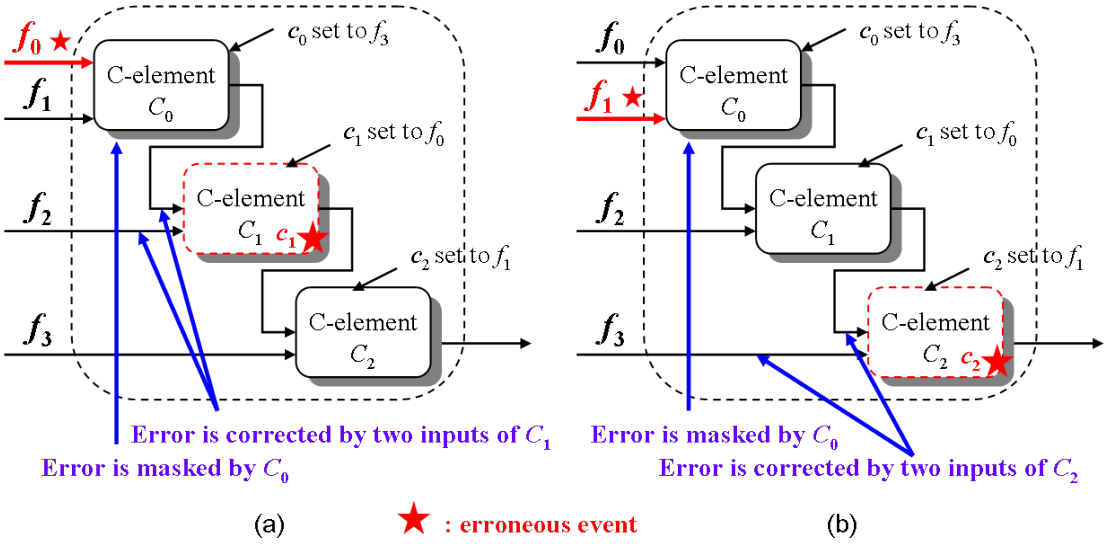


Figure 4.11: Two error-correction diagrams by using a single cascaded C-element set. (a) single-error event at x_i, (b) single-error event at f_k.

4.3.2.2 Double-Error Events

Double errors appear at f_k and f_l If a variable node v_i receives two erroneous messages at f_k and f_l, such that $|k-l| \geq 1$, there are three possible cases:

- If f_0 and f_1 are erroneous, the output of C_0 is thus erroneous. Since the rest of f_k and storages are correct, the error from C_0 is masked.
- f_0 and f_k $(k > 1)$ are incorrect. In this case, the output of C_0 is correct, and any erroneous f_k can't induce an error event.
- f_k and f_l, $(k \neq l \neq 0)$, are erroneous. In this case, any single error event is masked.

An example case is given in Fig. 4.12-(a). Both erroneous inputs, f_1 and f_3, are masked by C_0 and C_2, respectively.

Double errors appear at c_k and c_l If a variable node v_i generated two upsets at c_k and c_l, such that $|k-l| \geq 1$, these events can be sorted as a single error event occurrence. Consequently, these cases are masked by the inherent fault-tolerance by C-element.

For instance, an error-correction diagram is shown in Fig. 4.12-(b). Both false states, c_0 and c_2, can be corrected by two correct inputs of C_0 and C_2.

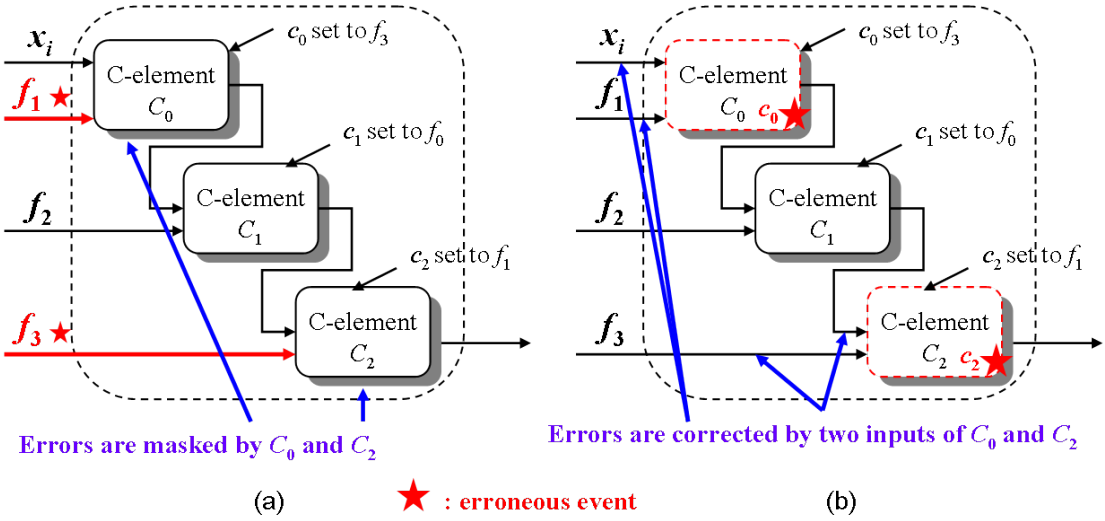


Figure 4.12: Two error-correction diagrams by using a single cascaded C-element set. (a) double-error event at f_k and f_l, (b) double-error event at c_k and c_l.

Double errors appear at f_k and c_l If two upsets occur at f_k and c_l of a variable node v_i, where $|k-l| \geq 0$, there are three possible cases:

- if $k = (d_v - 1)$ and $l \neq (d_v - 2)$, the value of C-element C_l, c_l, is correct. Since only single-error event is occurred among the C-elements except C_l, due to C-element's behavior another input of C_l is assured error-free. In this case, only f_k is erroneous that can not induce an error at the output.

- When $k \neq (d_v - 1)$ and $l \neq (d_v - 2)$, f_k and c_l are correct. Hence, the output from v_i is correct as well.

- At last, if $k = (d_v - 1)$ and $l = (d_v - 2)$, f_k and c_l are thus incorrect. In this case, the output is erroneous.

Fig. 4.13-(a) shows the error correction of double-error event at f_1 and c_1. The erroneous f_1, as one of the inputs of C_0 is masked by C_0. Furthermore, the state of C_1, c_1, is corrected by the two correct inputs.

However, the double-error pattern, the local incoming message to the last C-element and its storage, can not be corrected, as shown in Fig. 4.13-(b).

To sum up, a cascaded C-element set is able to correct any single error event regardless of where those errors originate. Any double-error events can be corrected, except a single pattern, namely the local incoming message to the last C-element and its storage are simultaneously flipped over.

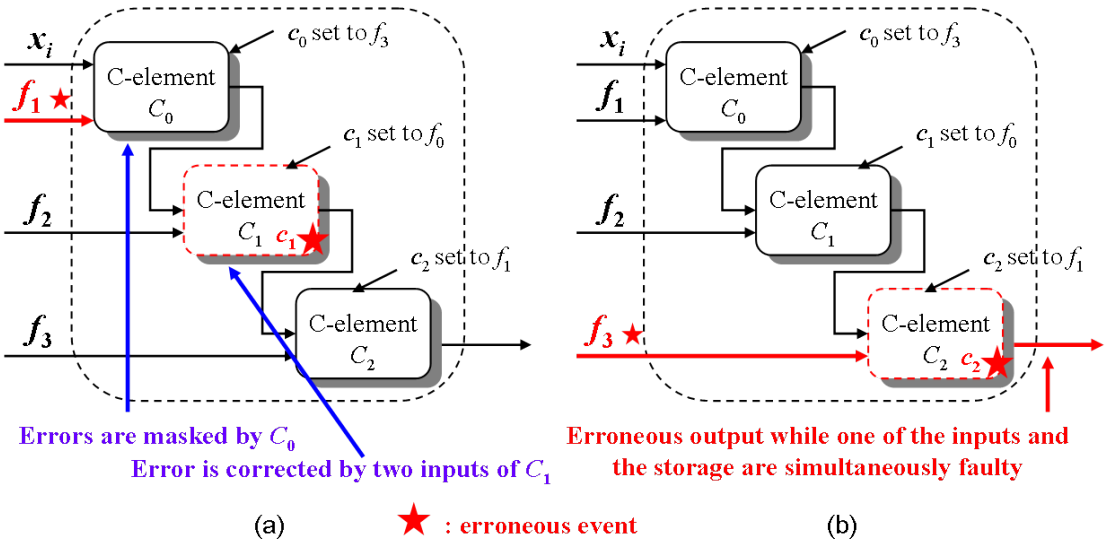


Figure 4.13: (a) Double-error event at f_k and c_l is corrected by using a single cascaded C-element set, (b) double-error event at f_3 and c_2 is the single double-error pattern that can not be corrected by a cascaded C-element set.

4.3.3 Fault-Tolerance Inherited by C-element

To further reveal the error-resilience inherited from C-element, we study the reliability of a state memory of C-element by comparison to a static logic gate, namely an inverter [TBW+13]. Without loss of generality, the state memory was designed for a 600nm CMOS logic process. For comparison, an inverter was also designed. Both circuits were simulated in Virtuoso Spectre from Cadence [Spe], where signal errors are set up in a large scale by the "noisescale" parameter. Fig. 4.14 shows an overlay of Monte Carlo transient simulation runs form both the state memory and the static gate simulations. An error occurs whenever the difference crosses $0.05V$ threshold when $5V$ as the correct output. More precisely, if the difference of output value to the correct

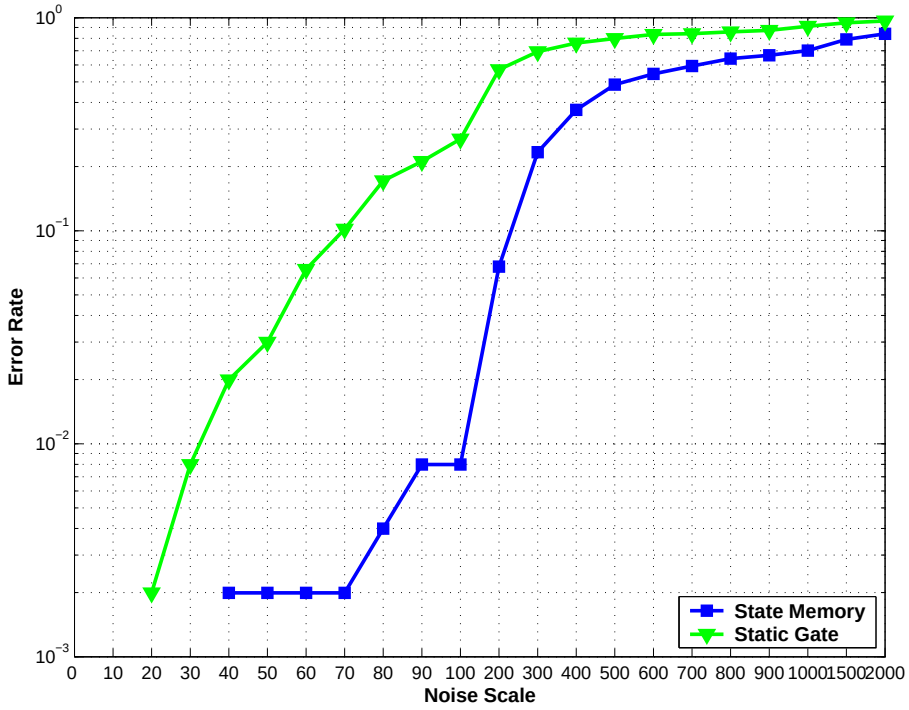


Figure 4.14: Simulated error rate results obtained using the 600nm CMOS model in Virtuoso Spectre. The "noisescale" parameter is set as large scale in order to induce enough upset cases to measure and compare the error rate for those two components.

one $(5V)$ is bigger than $0.05V$, then an error is considered. The difference lower than $0.05V$ is bypassed.

For the static gate, errors appear quite frequently as indicated by numerous threshold-crossings. By contrast, in the state memory case, the amplitude of output noise fluctuations is significantly cutoff. Thanks to the feedback mechanism in the state memory, up to a magnitude of two orders as the robustness gain as shown in Fig. 4.14. Consequently, by comparison to a static logic gate, C-element is able to tolerates the upset events with error-resilience gain between multiple times up to magnitude of two orders.

4.3.4 Application of MCD for a cDMR Model

In this subsection, we apply the MCD architecture to achieve fault-resilience in a digital logic architecture. The MCD algorithm is applied to a digital computation as shown in Fig. 4.15. A logic function $F(x)$ is implemented using a digital technology that is subject to errors at its output.

To evaluate the proposed architecture, a system composed of five systematic regular (4, 8) LDPC codes was simulated. All codes are a code rate 1/2. It means that there is one redundant parity bit for each systematic bit, i.e. $N = 2 \cdot K$. Moreover, the

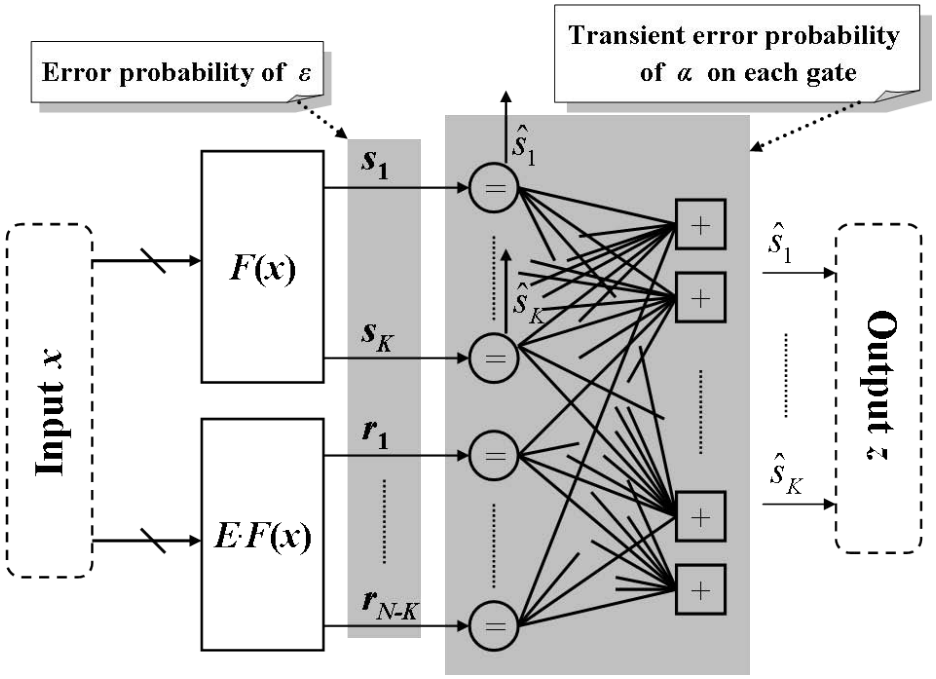


Figure 4.15: The proposed fault-tolerant architecture. The shaded area represents a MCD decoder implementation.

codeword sizes N are set as 64, 128, 256, 512, and 1024. Larger codes are expected to provide better error protection but are also more complex to integrate.

In our simulations, the output bits s and r from $F(x)$ and $E \cdot F(x)$, respectively, are assumed to have an uniform independent error probability of α. The XORs and C-elements operations that comprise the MCD architecture are assumed to be faulty boolean operations. Each gate in the MCD architecture has an uniform error probability of ε. To take into account of the impact of hard defects,"stuck-at" faults were also inserted in some uniformly random positions of $[s\ r]$ with an error rate of δ, $\delta = 0.001$ in practice.

The BER results for the LDPC codes associated MCD algorithm are shown in Fig. 4.16. As long as $\alpha < 0.05$, the output s from $F(x)$ can be recovered with a significantly low error probability. As α is reduced below δ, the performance becomes dominated by the gate-level fault probability ε. The simulation results show that, in the case of $\varepsilon = 10^{-5}$, the MCD architecture introduces gains about two orders of magnitude by comparison with uncoded data (output s from $F(x)$).

Consequently, the proposed decoding method is able to suppress the resulting error probability to a level equal to the decoder's internal fault rate.

For a given LDPC code with length N, the number of C-elements assigned for the design of the decoder is approximately $N \cdot d_v$, where d_v is the average variable node

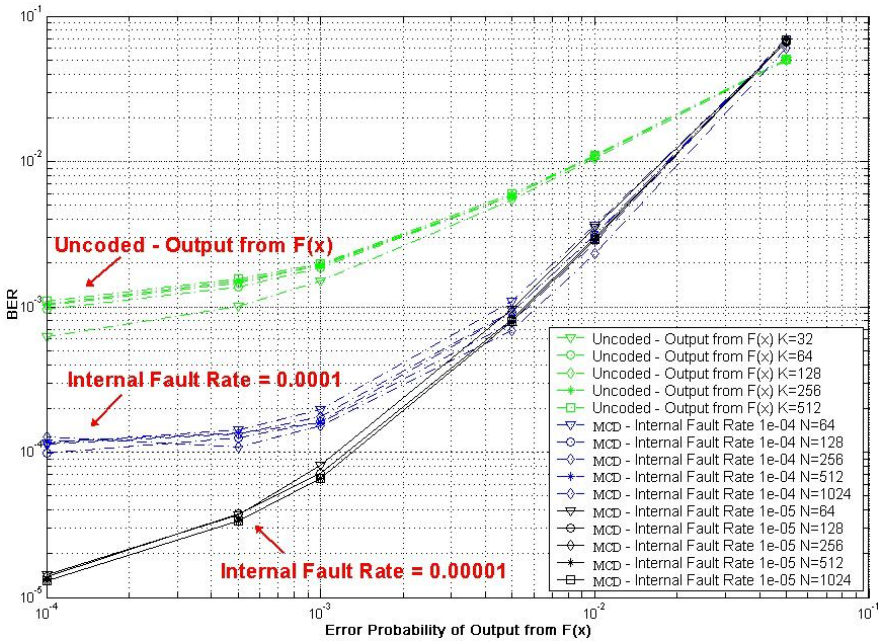


Figure 4.16: Simulation results for rate-1/2 LDPC codes based on MCD architecture with five iterations. The hard-fault rate δ is 0.001.

degree. The number of XOR gates is approximately $N \cdot R \cdot d_c$, where R is the code's rate ($R = 1/2$ in the experimentations), and d_c is the parity-check node degree. One of the considered codes is $N = 1024$, $R = 1/2$, $d_v = 4$ and $d_c = 8$. In this case the approximate total hardware complexity is equal to 8k logic gates.

One interesting result of our experimentations is that the MCD method provides similar performance across a wide range of code lengths. Error-correcting benefits may therefore be obtained for short codes, such as the $N = 64$ example. It is a length compatible with data words in current 64-bit processors. As suggested in [WH09], the MCD method can also achieve a reduced net redundancy if several parallel operations are packaged together, yielding a larger code size. In this "bundled-function" case, the relative hardware complexity of the decoder is reduced by comparison to the complexity of the functions themselves.

We also define the system's effective redundancy rate of hardware complexity R_E as follows. Let K_F be the number of logic gates for the implementation of $F(x)$. E_F is the number of gates necessary for the encoding function $E \cdot F(x)$. N_{MCD} is the number of gates of in the MCD module. Then, $R_E = (K_F + E_F + N_{MCD})/K_F$. According to the redundancy cost analysis in [WH09], the redundancy imposed by the LFCT is closed to K_F when $F(x)$ is an array of 5×5 combinational multipliers. In this work, N_{MCD} is considered as $N_{MCD} \approx K_F$ since the MCD is composed of XORs and C-elements as well as in the LFCT. Consequently, R_E is highly dependent on the gate complexity of $E \cdot F(x)$. Therefore, E_F is discussed as follow;

- In CMOS technology: It may be possible to perform logic minimization on $E \cdot F(x)$. If the number of parity bits is smaller than K, it is especially efficient. Moreover, if $F(x)$ is a boolean function, such like, a Finite State Machine (FSM) [Gil61]. Then, $E \cdot F(x)$ can be fabricated by a FSM as well. In this case, $E_F \approx K_F$.

- In hybrid CMOS/nanoelectronic technology [DL05]: A device with nanometer scale footprint that also is used in combination with a CMOS subsystem. Such hybrid circuits offer the potential for high defect tolerance. In this case, it is possible to tune redundancy cost to reliability for encoding function $E \cdot F(x)$ or $F(x)$.

- In nanoelectronic technology: Snider *et al.* described that nanowires based crossbars architecture that has capacity of fault-tolerance in [SKHW05]. An example of check symbol generator for self-checking has also been given, as well as the C-element implementation. Therefore, the proposed architecture would yield potential benefits in nanoelectornic devices.

To sum up, the redundancy rate of hardware complexity R_E depends on the fabrication technology and also the architecture. Typically, the cDMR model provides a robust architecture solution with redundancy of one in some cases, such like a FSM design in CMOS. Otherwise, due to the overhead redundancy imposed by the correlated-error-free function for $F(x)$ and $E \cdot F(x)$, R_E may be larger than one.

Further information for the generation of LDPC codes used in this work can be found in Annex. A.5. Moreover, the error-injection model for MCD method under a faulty decoding process is illustrated in Fig. 4.17. More details can be found in Section. 4.2.1.

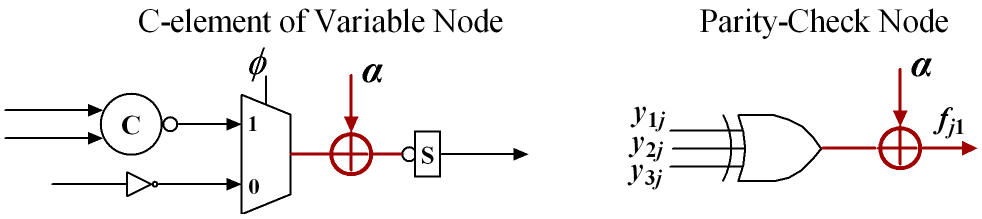


Figure 4.17: Noisy MCD architecture.

4.3.5 Modified MCD that Contains Feedback Mechanism

In order to further improve the capacity against internal transient faults, the MCD algorithm may be modified by adding a feedback connection. The feedback connection has been previously shown to suppress internal transient faults in [WLMT11]. Actually, the

output from a cascaded C-element set is fed into the input of this set. This modification is applied to Step 4 of the MCD algorithm shown in Section. 4.3.1:

1. Initialize $y_k = x_i$, for all $k \in \mathcal{V}$.
2. Compute $f_{ji} = \oplus_{m \in P_{j \setminus i}} y_{mj}$ for all $j \in \mathcal{P}$.
3. Initialize each C-element memory as $c_k = f_m$, where $m = (k + d_v - 1) \bmod d_v$.
4. The C-element's port connections are as follows. For C_0, the inputs are y_k and f_1, and the output is c_0. For the remaining C_k, the inputs are c_{k-1} and f_{k+1}, and the output is c_k.
5. Iterate steps 2 and 4 during a fixed number of iterations, as the restoration phase.
6. The corrected output is $z_i = c_{(d_v-1)}$.

This modification enables to improve the decoder's resilience to internal faults when time-redundancy is applied to the output bits, z_i. By comparison with the MCD, under a faulty process the gain in terms of decoding performance achieved by the feedback mechanism is shown in Fig. 4.31.

The MCD and MCDfb methods can be regarded as circuit-level designs for variable nodes in a LDPC decoder. Cascaded C-element sets are employed in both methods. In Fig. 4.18 and Fig. 4.19, the architectures of the variable nodes for MCD and MCDfb are detailed, respectively. During the decoding process, the modified C-element with phase operation is used to implement the algorithm's initialization phase. The feedback modification for the MCDfb method is also given in Fig. 4.19.

4.3.6 Decoding Method based on Three Input C-element

In this subsection, a decoding method based on a modified C-element with three inputs, referred as to MCD-3in, is presented. The decoding method keeps the same decoding process as well as the message exchanging rule to the MCD. But, the C-element unit contains a modified C-element with three inputs. This kind of decoding method is similar to the original Gallager-A decoding method. We first introduce the modified C-element with three inputs. Then, the decoding performance of MCD-3in is demonstrated.

4.3.6.1 Modified C-element with three input

The modified binary C-element with three inputs is shown in Fig. 4.20. In this achitecture, the "C-not" gate detects whether the three inputs are equal, with an output given by

$$Q = \begin{cases} \overline{A}, & \text{if } A == B == C \\ Z, & \text{otherwise,} \end{cases} \tag{4.2}$$

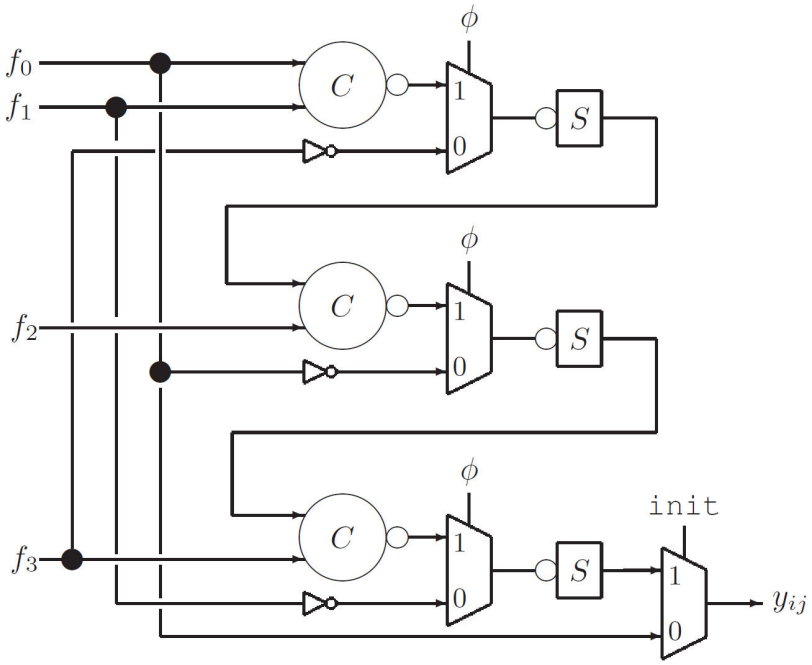


Figure 4.18: MCD architecture, $d_v = 4$.

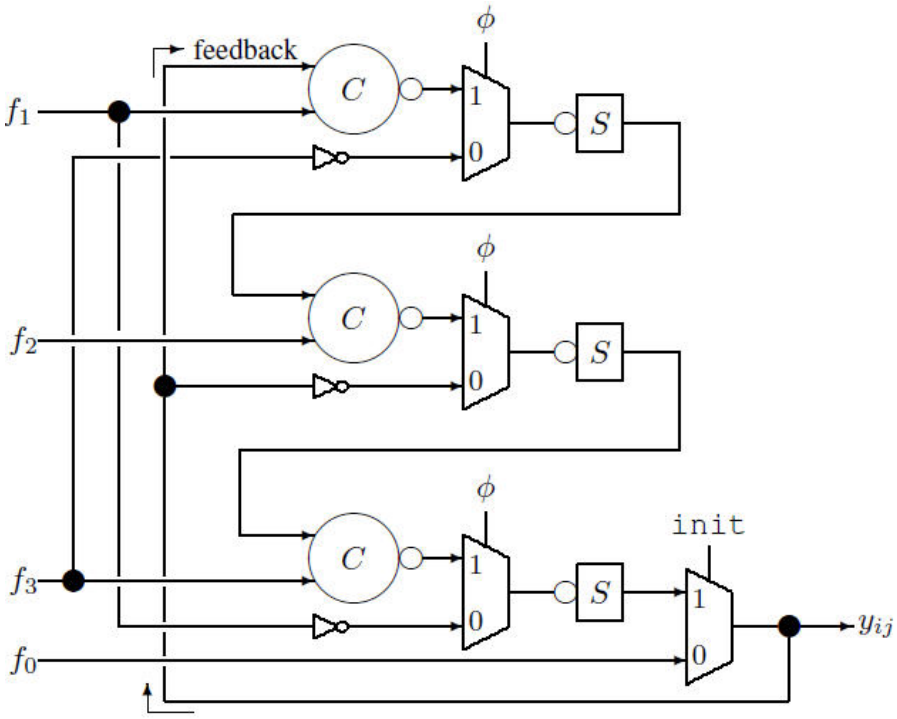


Figure 4.19: MCDfb architecture, $d_v = 4$.

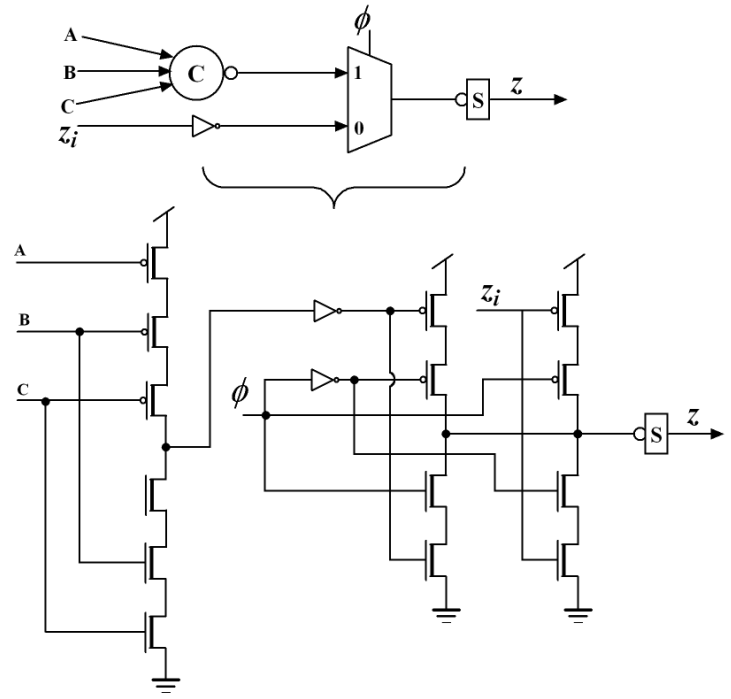


Figure 4.20: Modified C-element with three inputs. Pull-up and pull-down networks are built with three PMOS transistor and NMOS transistor, respectively.

where Z denotes a high-impedance output state. If $Q \neq Z$, then the C-not gate actively drives the S latch, overpowering its state. If $Q = Z$, then the state of S is maintained via weak feedback.

4.3.6.2 Decoding Method based on Three-Input C-element

By adapting the message passing rule of Gallager-A method, a MCD-3in approach can be regarded as an implementation of Gallager-A method by using modified C-element with three inputs. More precisely, in the case of $d_v = 4$ the MCD-3in algorithm can be described as follow:

1. Initialize $y_{ij} = x_i$ (the channel-side message), for all i.
2. Compute $f_{ji} = \oplus_{m \in P_{j \setminus i}} y_{mj}$ for all j.
3. For a given variable node v_i, d_v modified C-elements with three inputs are combined. Initialize each modified C-element's memory as x_i.
4. Let $f_1, \cdots, f_{d_v}$ the message from check nodes be the received messages in local for this node. The input of the first modified C-element are $f_2, \cdots, f_{d_v}$, and the second C-element has as inputs $f_1, f_3, \cdots, f_{d_v}$, and etc. The output of variable node is the output of each modified C-element.

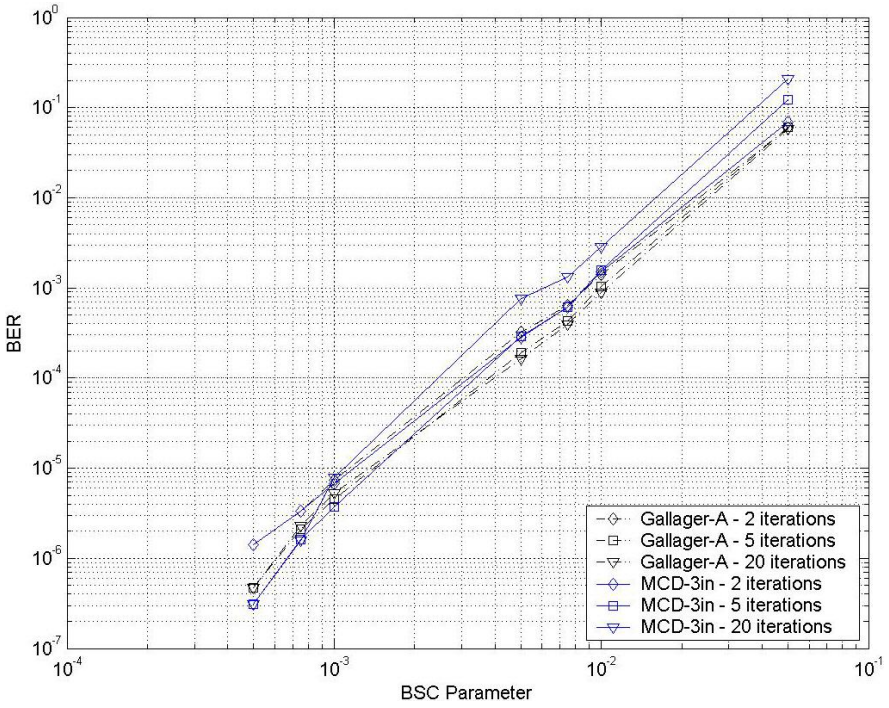


Figure 4.21: Simulation of (4,8) LDPC codes of length 64 for Gallager-A and MCD-3in decoding methods, under an error-free decoding process.

5. Iterate steps 2 and 4 during a fixed number of iterations.

6. The corrected output z_i is the output of a variable node.

In fact, the MCD-3in is similar to the Gallager-A method in the case of $d_v = 4$. To take into account of different degree of a variable node, the decoding method can be modified in steps 3 and 4. The number of input of corresponding modified C-element should be concerned to the number of incoming message from check node. For instance, when $d_v = 5$, the corresponding C-element should be built from four inputs. The architecture of a four-input C-element is similar to the modified C-element with three inputs as shown in Fig. 4.20. The only difference is the number of input to the C-element that is equal to the number of transistors assigned to pull-up and pull-down networks.

In addition, another alternative architecture of variable node $d_v = 5$ is demonstrated in Fig. 4.22. In this architecture, two C-not components and a C-element are selected to carry out the operation of a variable node. More precisely, four incoming messages from check nodes are first fed into two parallel C-nots. After a C-element takes the outputs of two C-nots as its inputs. The output of the variable node is the state of C-element. According to [Bak10], the more transistors are used in serial way, such like the pull-up/pull-down network in Fig. 4.20, the larger latency would be in-

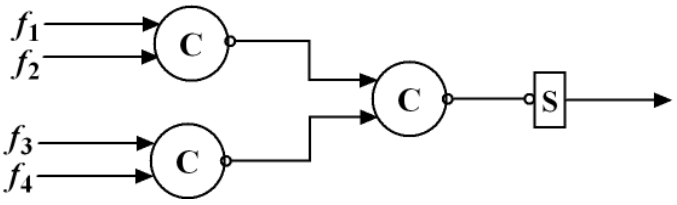


Figure 4.22: Alternative architecture of a variable node $d_v = 5$.

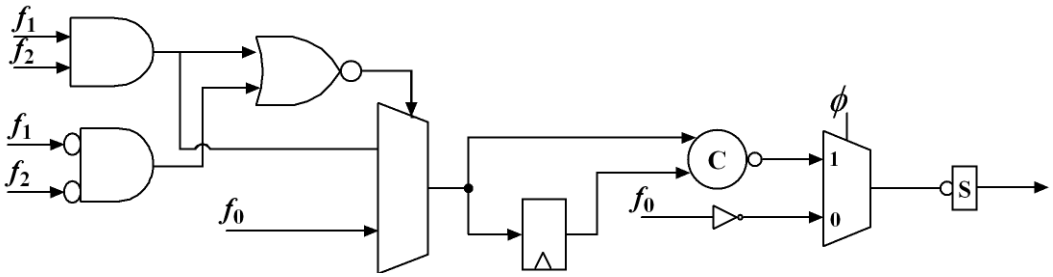


Figure 4.23: Architecture of a variable node of flip-flop based C-element decoding method.

duced due to the propagation timing through transistors. In this case, the architecture in Fig. 4.22 requires a bit more hardware complexity. But it reduces the processing delay.

LDPC codes of length 64 have been simulated with a Gallager-A and MCD-3in under an error-free decoding process, as shown in Fig. 4.21. As a result, the curves of BER performance by those two decoding methods are almost matched. As the study of error-resilience study shown in Section. 4.3.3, thanks to the feedback mechanism provided by the state memory, more benefits can be achieved against transient faults. In this case, the MCD-3in would be more efficient for the error-correction in the presence of internal transient faults.

4.3.7 Flip-Flop Based C-element Decoding Method

The modified C-element based decoding method presented in Section. 4.3.6 achieves similar error-correction performance as to the GBF. But it costs less hardware complexity by comparison to the GBF. In this subsection, we introduce a flip-flop based C-element decoding method. This flip-flop based C-element decoding method is originated from the Gallager-A decoding method, as a variant of Gallager-A algorithm.

By keeping the same process for the check nodes as the Gallager-A, the flip-flop based C-element adds an additional flip-flop and also a C-element to the structure of a Gallager-A variable node. Fig. 4.23 shows the architecture of a variable node of flip-flop based C-element decoding method. More precisely, the incoming messages from check nodes f_1 and f_2 are processed by a Gallager-A method's structure as introduced

in Section. 3.2.5. The output from multiplexer is then sent to a C-element and also restored in a flip-flop. Another input of the C-element is taken from the output of the flip-flop. In addition, the state of the C-element is initialized as the received value f_0. Such that, Table. 4.1 illustrates the message-processing diagram for the variable node by order of decoding iterations.

Iteration	f_0	f_1	f_2	Flip-Flop	Output
0	0	1	0	$'X'$	0
1	0	1	0	0	0
2	0	1	1	1	0
3	0	1	1	1	1
4	0	0	1	0	1
5	0	1	1	1	1
6	0	0	1	0	1
7	0	0	1	0	0
8	0	0	0	0	0

Table 4.1: The message-processing diagram for the variable node of flip-flop based C-element decoding method, by order of decoding iterations. Moreover, $'X'$ represents the unknown value.

Fig. 4.24 reveals the gain in terms of BER performance achieved by the flip-flop based C-element method by comparison to the Gallager-A method. The noisy model of the flip-flop based C-element method is demonstrated in Fig. 4.25 and the noisy model for Gallager-A is the one of Fig. 4.8. Actually, better BER performance is obtained when more decoding iterations are allowed for the flip-flop based C-element. Furthermore, it yields a gain in terms of decoding performance close to two orders of magnitude. However, from Fig. 4.24, the BER performance is worsened by an increase of number of iteration. For instance, the BER result of 20 iterations is worse then the one of 5 iterations.

In this case, we further analyze its property with increasing the number of decoding iterations. In Fig. 4.26, it is shown that the BER performance of flip-flop based C-element method goes up and down by decoding iterations. When the error rate of internal faults is 10^{-4}, it floats from 2×10^{-4} to 0.5×10^{-4}. More precisely, in some iterations the decoding method is able to render an interest of error-correction from the internal upsets. For the rest, the decoding method is not helpful. Consequently, the flip-flop based C-element decoding method can not be stabilized by an increase of number of decoding iterations.

Remark 3. *Significantly, under a faulty decoding process, the MCD provides the stability when the number of decoding iterations increases. Moreover, the MCD achieves a property of fast-convergences. We further study the properties of MCD in Section. 4.5.*

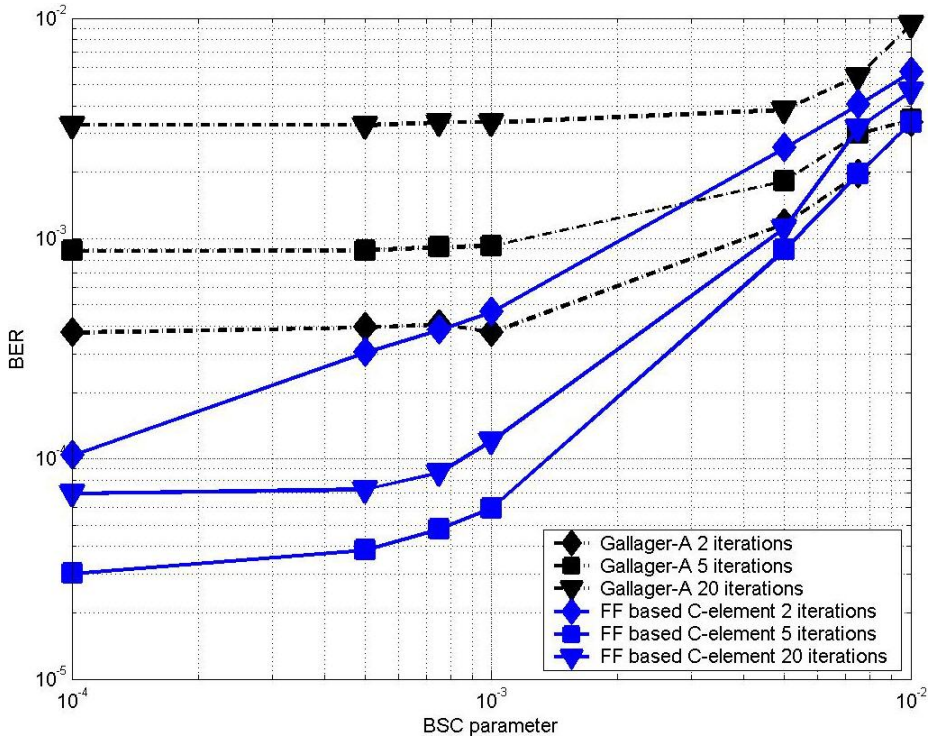


Figure 4.24: (3,6) LDPC codes of length 64 are simulated for Gallager-A decoding method and the flip-flop based C-element decoding method as well. For the error-injection of internal faults, the error rate α is 10^{-4}.

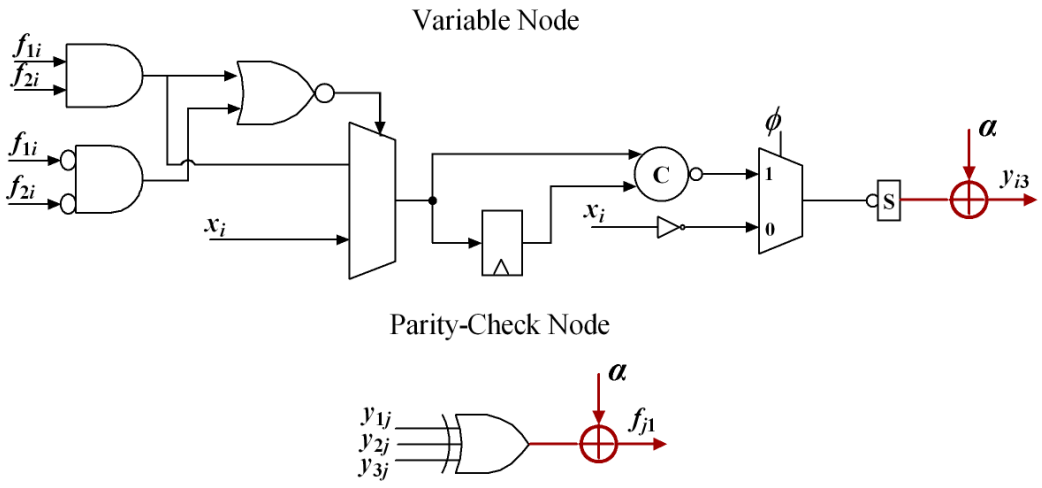


Figure 4.25: Noisy model for flip-flop based C-element decoding method

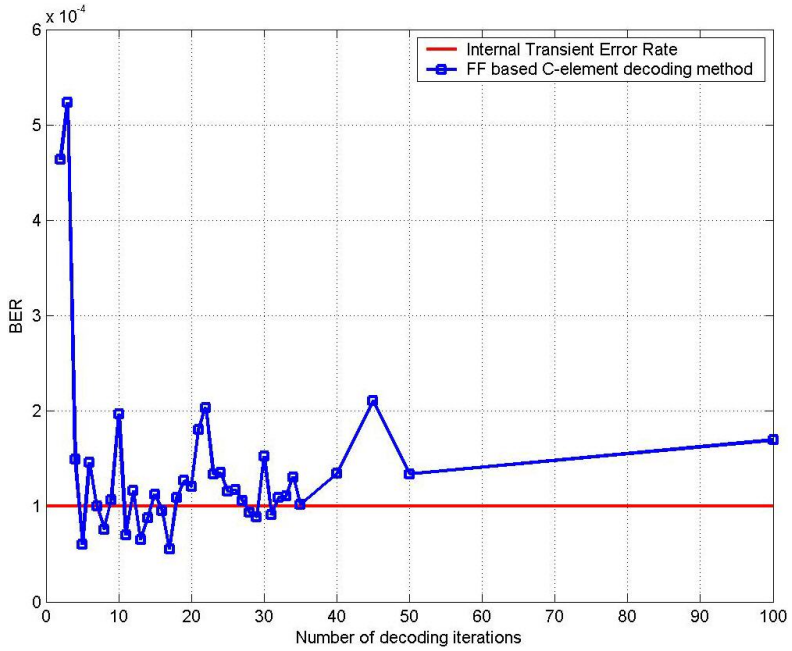


Figure 4.26: (3,6) LDPC codes of length 64 are simulated for flip-flop based C-element method with internal error-injection of error rate $\alpha = 10^{-4}$. Moreover, the BSC parameter ε is 10^{-3}.

4.4 The Space-Time Redundancy Technique

All the presented BER performance under a faulty decoding process by using GBF and MCD (see Fig. 4.16), can not beat the level of internal error-rate. In this section, in order to further improve the decoder's resilience to internal faults, we propose a low-redundancy fault-tolerant technique that is able to considerably increases decoder's decoding performance in the case of transient faults within decoder itself. Significantly, it even can improve the error-correction capacity of decoder when the internal fault rate is as high as 0.01.

4.4.1 The Majority Mechanism

In fact, the Space-Time Redundancy technique is done by a temporal majority mechanism that is applied at the decoder's output. The temporal-redundancy strategy is expected to improve fault correction because the MCD and MCDfb methods are derived from stochastic decoders, in which messages are considered as stochastic *streams* [GR03, WGRS05]. In a stochastic decoder, the output decisions are rendered by performing a time-majority on the output streams. Since the MCD and MCDfb decoders are already performing iterations, the time-voting can occur during the last few iterations.

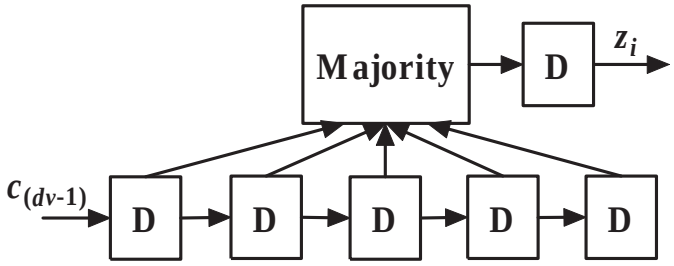


Figure 4.27: Structure for a temporal three-of-five majority voter.

To validate the effect of temporal redundancy, the decoder's output from $c_{(d_v-1)}$ are processed into a majority operation, $\mathrm{Maj}(c_{(d_v-1)}, l)$ where l denotes the number of samples. We consider a three-of-five voter (e.g. $l = 5$) as shown in Fig. 4.27.

4.4.2 On the Decoding Performances Impact of a Space-Time Technique

In this subsection, three decoding methods, Gallager-A, MCD, and MCDfb, are applied to the cDMR model as the ECC engine (see Fig. 4.15 in Section 4.3.4).

The Gallager-A, MCD and MCDfb methods were simulated using a systematic regular (3, 6) LDPC code. To be compliant with typical word sizes used in electronic devices, the codeword size is $N = 64$, corresponding to 32-bit data words. In our simulations, the output bits s and r from $F(x)$ and $E \cdot F(x)$, respectively, are assumed to have an uniform independent error probability of ε. The logic gates that are contained in the decoding circuit are assumed to be faulty boolean operations, which have a uniform error probability of α. In our simulations, the temporal-majority component is assumed to be a reliable output interface circuit.

The BER performance results with no time-redundancy are shown in Fig. 4.28 and Fig. 4.29. In this context, all methods achieve poor performance. In the high-α case shown in Fig 4.29, the Gallager-A method degrades completely with an increase of number of iterations. The results using time-redundancy are shown in Fig. 4.30 and Fig. 4.31. Unlike the degradation with increased iterations in Gallager-A, the MCD methods introduce a significant improvement in terms of BER performance for the high-α case.

Time-redundancy is not effective when it is applied to the traditional Gallager-A decoder. Indeed, its degradations are too severe. In addition, the feedback approach is shown to be beneficial in the high-α case as well.

As a conclusion, the achievements obtained by MCD methods and Space-time technique are summarized as follow:

1. When the intrinsic gate-error α is high, the Gallager-A performance worsens with increased iterations, but the MCD methods do not exhibit this degradation.

2. The feedback approach is shown to be beneficial in the high-$alpha$ case as well.

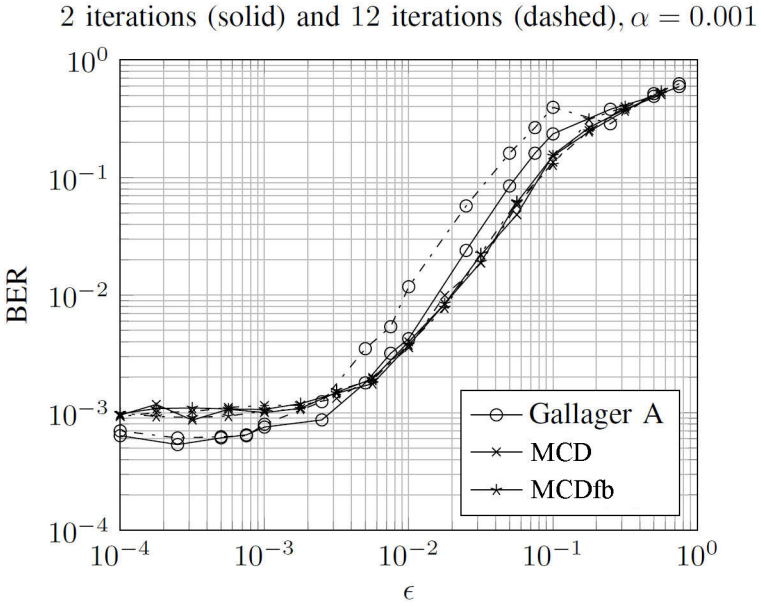


Figure 4.28: BER results versus iteration number without applying time-redundancy.

3. Once the time-redundancy is applied at the decoder's output, one or more orders of magnitude gain is achieved.

4. Consequently, MCD and MCDfb with apply time-redundancy yield improved fault-tolerant potential against internal upsets, even with a high gate-error rate.

4.4.3 A feasible time-redundancy implementation

In order to demonstrate the benefits of time-redundancy in the MCD kind decoders, it is necessary to assume that the time-voting module can be implemented with error-free components. This assumption has to be analyszed. In the study of "post-CMOS" nano-scale electronics, it is common to imagine a hybrid system in which logic operations are built from faulty nano-scale gates, while output interfaces are implemented using larger more reliable CMOS devices. For any fault-masking method, it is critical to assume that the final read-out circuits are more reliable than the other internal components, or else the system's overall reliability will be limited by the interfaces themselves. It is furthermore possible to imagine alternative architectural solutions for implementing time-redundancy. For example, if the design technology operates at a much higher speed than the output interface devices, then the time-redundancy may be implemented via passive RC filtering in the output interface. An example is shown in Fig. 4.32, where low-pass filtering is performed by parasitic elements within two inverter buffers. This arrangement achieves an average-and-threshold operation, which is functionally equivalent to a time-voting mechanism used in our simulations.

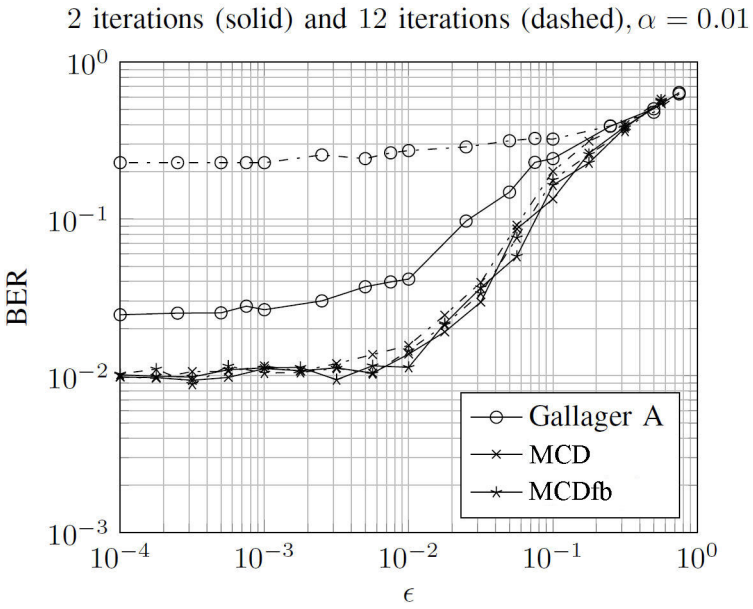


Figure 4.29: BER results versus iteration number without applying time-redundancy. When the intrinsic gate-error rate (α) is high, the Gallager-A performance worsens with increased iterations. The MCD methods do not exhibit this degradation.

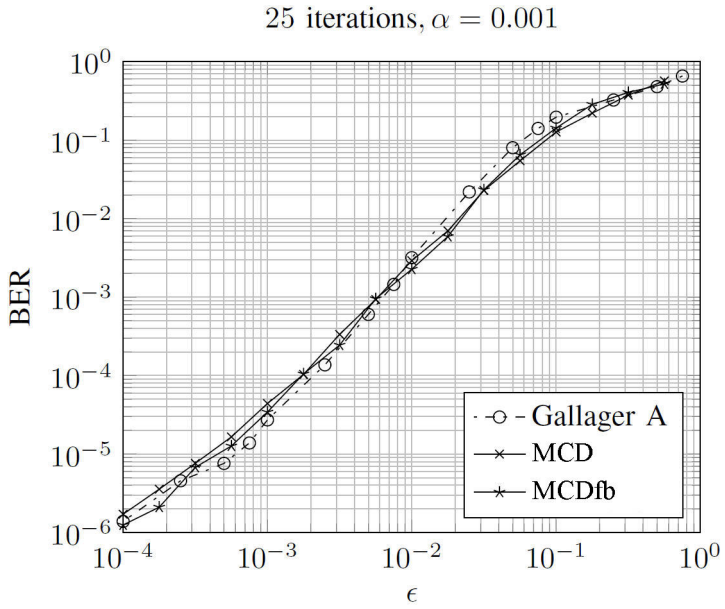


Figure 4.30: BER performance results with time-redundancy applied at the decoder's output.

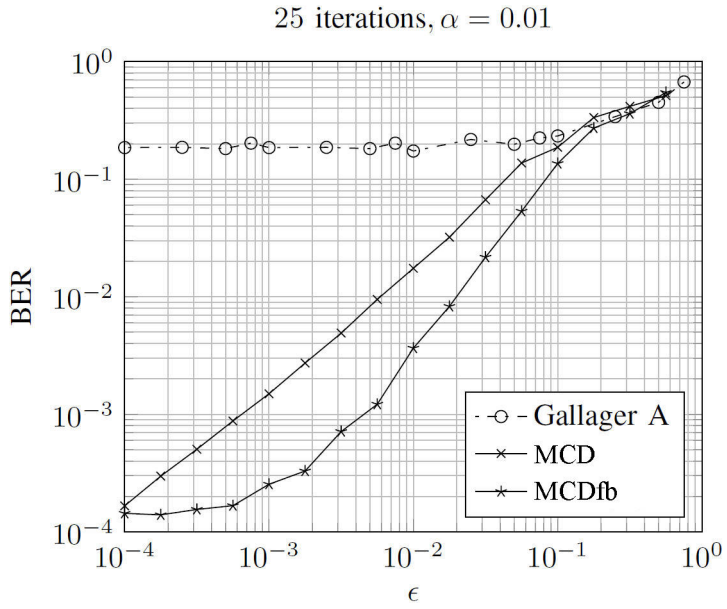


Figure 4.31: BER performance results with time-redundancy applied at the decoder's output. The MCD methods achieve improved performance when α is high. The feedback approach is also shown to be beneficial in the high-α case.

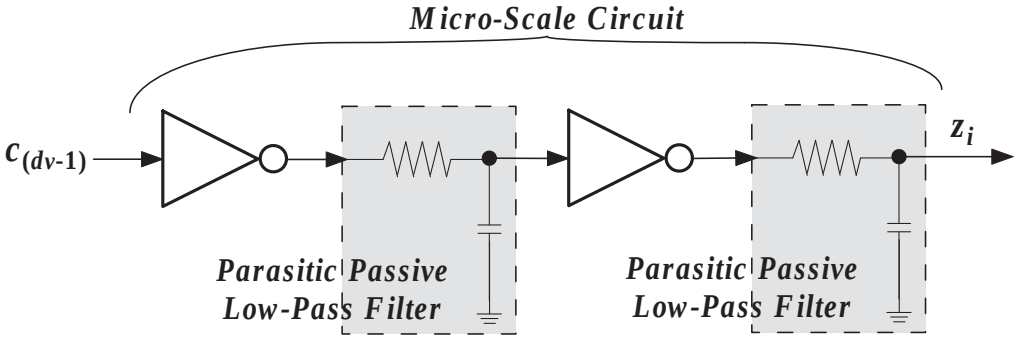


Figure 4.32: Passive time-redundancy obtained by a low-pass filter. This approach may be implemented, for instance, with a native R-C parasitics in a large-size output buffer.

4.5 Further Analyses on MCD

In this section, we first investigate the threshold determinations for MCD and GBF with the Density Evolution approach. By means originated from *Trellis Structure* [Ung82], a further detailed analysis for MCD is also illustrated. In addition, a rough estimation of dynamic power for MCD is given. Finally, a good candidate for the ECC of cDMR is proposed.

4.5.1 An Approximation of Threshold Determinations

In this paragraph, the performances of GBF and MCD are estimated under an iterative faulty decoding process by applying an error model. Due to the effects of state memory from C-elements in MCD case, its capacity can not be resolved by thoroughly adapting the *Density Evolution* derivations. Therefore, we give out an approximation for the threshold determination of MCD. Hence, the estimation of decoding performance under a faulty process for MCD is also an approximation. More precisely, we assume the probability of variable nodes do not depend on the messages from check nodes any more after the first iteration of restoration phase.

4.5.1.1 GBF Decoder's Performance under Faulty Decoding

Firstly, we start with the estimation of GBF decoder's capacity under Faulty Decoding. The performances of LDPC codes for noisy decoders, such like the *Gallager-A* decoder, have been discussed in [Var11]. In this work, we further study the capacity of GBF decoding method by adapting the same error model than in [Var11], as shown in Fig. 4.6.

For the check node operation, the outgoing message is incorrect in two cases. An odd number of the messages from variable nodes are erroneous and XOR itself operates rightly. No odd number of the messages are in error and XOR is faulty. Let $Px_1^{(l)}$ and $Px_{-1}^{(l)}$ be the probabilities that the XOR gives a correct result and incorrect one, respectively. They can be expressed as:

$$Px_1^{(l)} = 1 - Px_{-1}^{(l)} = \frac{1 + \left(1 - 2Pb_{-1}^{(l-1)}\right)^{(d_c-1)}}{2} \tag{4.3}$$

Thus, $q_{-1}^{(l)}$ can be obtained as

$$q_{-1}^{(l)} = Px_{-1}^{(l)}(1-\beta) + \beta Px_1^{(l)} \tag{4.4}$$

and $q_1^{(l)} = 1 - q_{-1}^{(l)}$.

For variable node's error probability, three events can lead variable node's value in error. The first is that the received value x_i is incorrect when less than b correct incoming messages f_{ji} are taken. The second is that at least b erroneous messages f_{ji}

disagrees with x_i. The last is that an internal fault occurs. More precisely, the event ω_-, at least b erroneous messages disagree with x_i and the event ω_+, more than b correct incoming messages disagree with x_i are expressed as:

$$\omega_- = \sum_{i=b}^{d_v-1} \binom{d_v-1}{i} (q_{-1}^{(l-1)})^i (q_1^{(l-1)})^{d_v-1-i} \tag{4.5}$$

$$\omega_+ = \sum_{i=b}^{d_v-1} \binom{d_v-1}{i} (q_1^{(l-1)})^i (q_{-1}^{(l-1)})^{d_v-1-i} \tag{4.6}$$

Due to the affection by BSC with parameter α, let $P_G^{(l)}$ represents $Pb_{-1}^{(l)}$ for GBF algorithm. This expression can be derived as following:

$$P_G^{(l)} = \alpha + (\varepsilon(1-\omega_+) + \omega_-(1-\varepsilon))(1-2\alpha) \tag{4.7}$$

4.5.1.2 MCD Performance under Faulty Decoding

The MCD performs two phase decoding process in the variable nodes. The MCD performance is thus analyzed in two phases. First consider the initialization phase in which the cascaded C-elements' state memories are set to $f_{k'}$, as detailed at step 3) in 4.3.1. To obtain the error probability of a variable node after the first iteration $P_E^{(1)}$, the error probability of each C-element memory c_k need to be considered at first. For instance, if $d_v = 4$, let τ_k represents the error probability of each C-element memory c_k. As depict in Fig. 4.10, τ_0 is incorrect in two cases: not both of f_0 and f_1 are correct when f_3 is in error; or f_0 and f_1 are both erroneous while f_3 is correct, thus,

$$\tau_0 = \varepsilon(1-(q_1^{(1)})^2) + (q_{-1}^{(1)})^2(1-\varepsilon),$$

Likewise, τ_1 and τ_2 are

$$\tau_1 = \varepsilon(1-(q_1^{(1)})^3) + (q_{-1}^{(1)})^3(1-\varepsilon),$$

$$\tau_2 = \varepsilon(1-(q_1^{(1)})^2-(q_1^{(1)})^3) + (1-\varepsilon)((q_{-1}^{(1)})^2 + (q_{-1}^{(1)})^3).$$

Moreover, the last C-element's state memory is the value of variable node and the outgoing message as well, $P_E^{(1)} = \tau_2$. To take into account the presence of internal upsets with error probability α, the probability of a variable node during the initialization step is thus modified as

$$P_E^{(1)} = \alpha(1-\tau_2) + \tau_2(1-\alpha) \tag{4.8}$$

As the check node, $q_{-1}^{(l)}$ and $q_1^{(l)}$ keep the same expression as (4.4), when $l = 1$.

During the second phase of the decoding process, please note that the variable node relies on its proper belief since C-elements carry the memories along iterations. Thus,

the incoming messages f_{ji} are not propagated further among nodes. In this case, the error probability $Px_{-1}^{(l)}$ for the MCD architecture $qx_{-1}^{(l)}$ is constant

$$qx_1^{(l)} = 1 - qx_{-1}^{(l)} = \frac{1 + \left(1 - 2P_E^{(1)}\right)^{(d_c - 1)}}{2} \tag{4.9}$$

Hence, $q_{-1}^{(l)}$ for the MCD $qe_{-1}^{(l)}$ is

$$qe_{-1}^{(l)} = qx_{-1}^{(l)}(1 - \beta) + \beta qx_1^{(l)} \tag{4.10}$$

For the variable node, the error probability of state memory at l^{th} iteration $\check{\tau}_k^{(l)}$ is obtained by a recursive process

$$\check{\tau}_k^{(l)} = \check{\tau}_{k-1}^{(l)} qe_{-1}^{(l)} + \tau_k(\check{\tau}_{k-1}^{(l)} qe_1^{(l)} + qe_{-1}^{(l)}(1 - \check{\tau}_{k-1}^{(l)})),$$

where $\check{\tau}_0^{(l)} = \varepsilon qe_{-1}^{(l)} + \tau_0(\varepsilon qe_1^{(l)} + qe_{-1}^{(l)}(1-\varepsilon))$. Sequentially, the density evolution equation for MCD decoder is deducted by a recursive expression:

$$P_E^{(l)} = \alpha + \check{\tau}_{d_v-2}^{(l)}(1 - 2\alpha). \tag{4.11}$$

4.5.1.3 Decoding Performance Estimations

By applying the derivations obtained in previous paragraphs, we illustrate the decoding performance estimations in the case of error-free decoding and faulty decoding processes for MCD and GBF decoders. Please note that the estimations for MCD are approximations.

For a threshold analysis, with (4.7) and (4.11), the maximal channel parameters ε^* for GBF and MCD are shown in Fig. 4.34 in the case of cycle-free and error-free, namely $\alpha = \beta = 0$. Although the difference of threshold decreases when the degree of the check node increases, the GBF has a larger decoding capacity overall in the case of error-free architecture. On the contrary, when the architecture is unreliable, the proposed MCD against internal faults significantly increases the maximal value ε^* and the internal error rate as well. As detailed in Table. 4.2, LDPC codes are simulated under a stop condition, benefit from decoder [Var11], namely $Pb_{-1}^{(l)} < \varepsilon$. For (3, 6) LDPC codes, MCD is able to process internal error rate as high as 0.0303, but only 0.0082 for the GBF approach. For (3, 12) LDPC codes, the thresholds of channel parameter and internal error rate for the MCD are 0.0096 and 0.0199. Those maximum values of parameters are even higher for the ones of the GBF approach in the case of (3, 6) LDPC codes.

4.5.2 Trellis Structure Like Analysis for MCD

Based on *Trellis Structure* [Ung82], a further detailed analysis for MCD is illustrated in this subsection. The *Trellis Structure* that was first introduced by Ungerboeck in

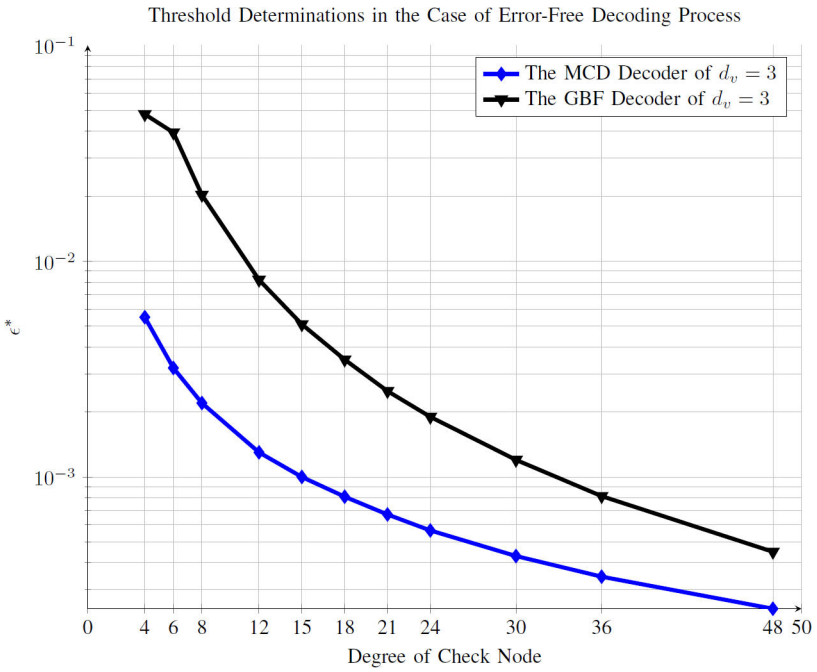


Figure 4.33: The threshold determinations over BSC for GBF and MCD decoders in the cases of cycle-free and error-free decoding process, in function of parity-check node's degree.

	GBF	MCD			
	(d_v,d_c)	(d_v,d_c)	(d_v,d_c)	(d_v,d_c)	(d_v,d_c)
	(3,6)	(3,6)	(3,12)	(3,24)	(3,48)
α^*	0.0082	0.0303	0.0096	0.0032	0.0011
ε^*	0.0126	0.0690	0.0199	0.0059	0.0011

Table 4.2: Maximal values of parameter α^* and ε^* that are determined under the stop condition, $Pb_{-1}^{(l)} < \varepsilon$ ($\beta = \alpha$).

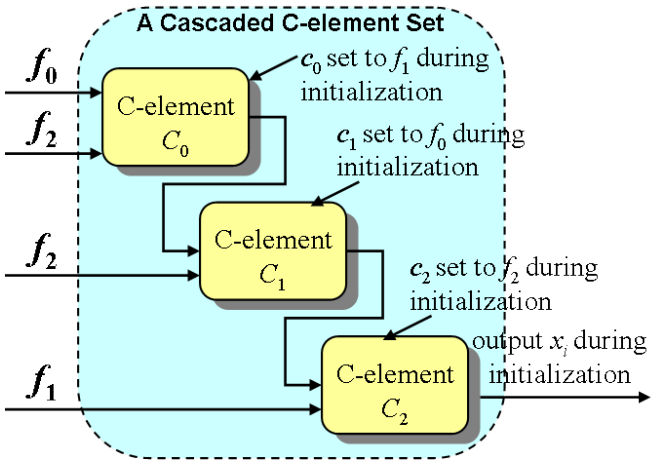


Figure 4.34: Implementation of a variable node, $d_v = 3$.

1982 [Ung82], is able to assign the state transitions of some channel codes, such like convolutional codes. Furthermore, by depicting codes' state transitions in a trellis structure, it is easily to improve its error-correction performance, e.g., Viterbi Maximum Likelihood (ML) decoding.

The rest of this subsection is organized as following. The trellis structure of $d_v = 3$ MCD is first presented. Then two succinct state transition diagrams for C-element's state memories are revealed in two cases: when the received channel message is correct and when the received message is incorrect. Afterwards results of two Monte Carlo simulations are shown to observe the states of C-element memory in function of channel parameter ε, in the cases of error-free decoding process and noisy decoding process. At last, some MCD's decoding properties are pointed out.

4.5.2.1 Trellis Structure for a MCD code

The error-correcting capacity of MCD was analyzed in previous subsection, but by approximating approaches. Due to the effects of C-element's state memory in a MCD variable node, the *Density Evolution* that is supposed to be the good metric to solve decoder's threshold by given some conditions or constraints can not be adapted in our context.

For a sake of facility, we study the case of $d_v = 3$. A single $d_v = 3$ MCD variable architecture, that contains a cascaded C-elements, is given in Fig. 4.34. When $d_v = 3$, the connection among cascaded C-elements is particular, not like specified in Algorithm. 4.3.1. More precisely, in order to reproduce more information from the incoming message of check nodes, a dummy signal, such as the f_2 reduplicated once in Fig. 4.34.

To take into account of the trellis structure for the MCD, we study the state transitions of a single architecture of a variable node, as shown in Fig. 4.34. In practice,

we first illustrate all the combinations of the state memories as the start state. By summarizing the end state after the execution of cascaded C-elements, the trellis structure is obtained. The trellis structure that assigns the state transitions for a $d_v = 3$ MCD is shown in Fig. 4.35. The structure is presented in two cases: when the channel message is correct and incorrect. Since the output of a variable node depends on C-element's state memory, we depict all the patterns of a memory state. When the number of state memory is three, eight combinations are obtained as shown in Fig. 4.35.

Moreover, three binaries that are filled in a shaped square stands for three state memories' values independently, e.g., '101' represents: the state of the first memory c_0 is '1', the state of the second memory c_1 is '0', and the state of the last one c_2 is '1'. Please note that when the channel message f_0 is correct, f_0 ='0'. Otherwise, f_0 ='1'. In Fig. 4.35, the state transition for variable node's output is evolved by incoming messages from check nodes, i.e., f_1 and f_2. As such, four patterns, "00","01","10", and "11", are the combinations of incoming messages.

4.5.2.2 State Transition Diagrams

In order to simplify the state transitions of the MCD algorithm, two simplified diagrams are presented in Fig. 4.36 and 4.37. Likewise, the two figures represents for the transitions when the channel message is correct and incorrect, respectively.

At first, when the channel message is correct, the state transition diagram is shown in Fig. 4.36. By varying the incoming messages f_1 and f_2, the state memory can be summarized into four cases. The first case is "001". The second case is "000". The third case is that the other three states of variable node output is 0 as a correct one (namely "010", "100", and "110"). The last case is that the other three states of variable node output is 1 as erroneous (namely "101", "011", and "111"). **As a result, when the channel message is correct the state of memory easily ends up with "000" as a correct output. Furthermore, when it arrives at "000", it can not be shifted.**

Subsequently, when the channel message is incorrect, the state transition diagram is introduced in Fig. 4.37. Similarly, the state memory can be summarized into four cases. The first case is "110". The second case is "111". The third case is that the other three states of variable node output is 0 as a correct one (namely "000", "010", and "100"). The last case is that the other three states of variable node output is 1 as erroneous (namely "110", "011", and "001"). **By contrast, if f_0 is incorrect, it turns out that the state of memory is contrarily stabilized at "111" as an incorrect output.**

4.5.2.3 Monte Carlo Simulations of Memory State

To take into account the state transitions of C-element memory in function of the channel parameter, (3,6) LDPC codes of length 64 are simulated in two cases: error-free decoding process and noisy decoding process. The results are given in Fig. 4.38 and Fig. 4.39, respectively.

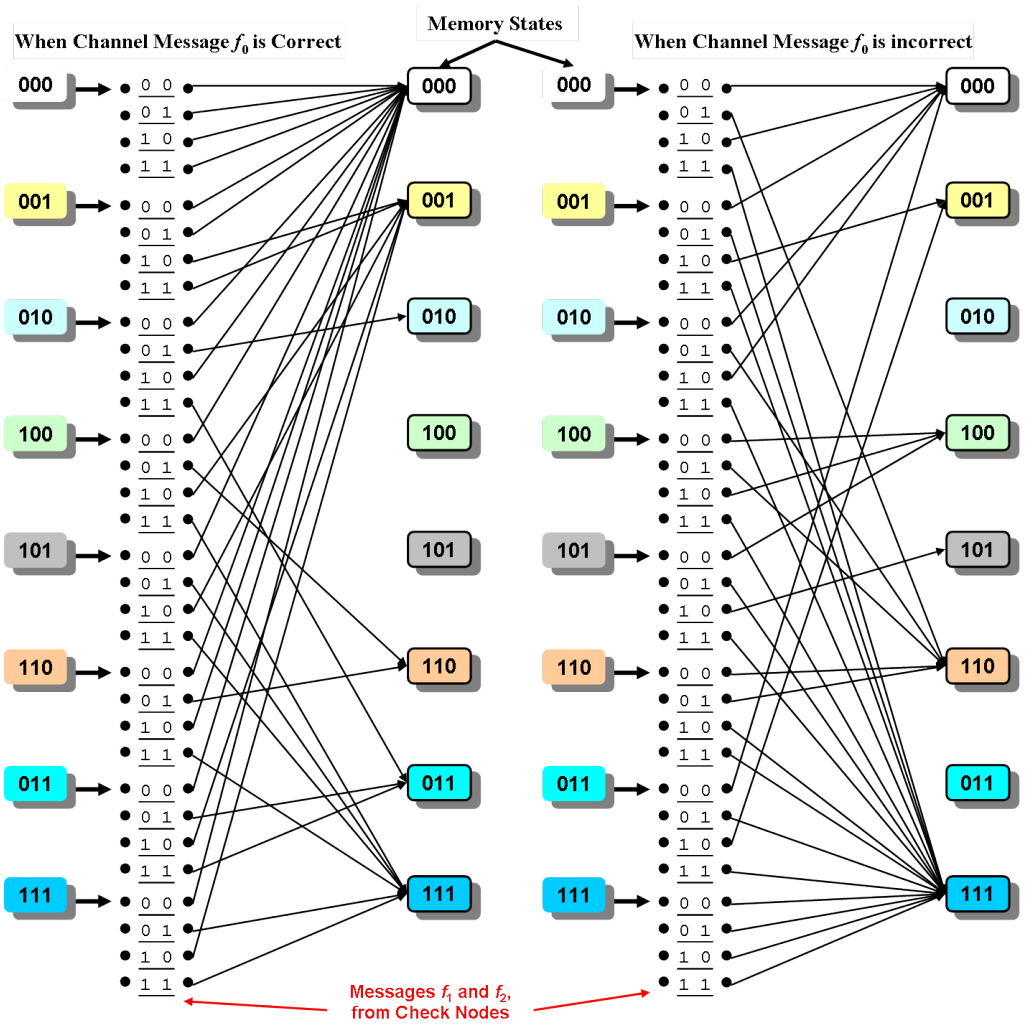


Figure 4.35: Trellis structure for a $d_v = 3$ MCD code.

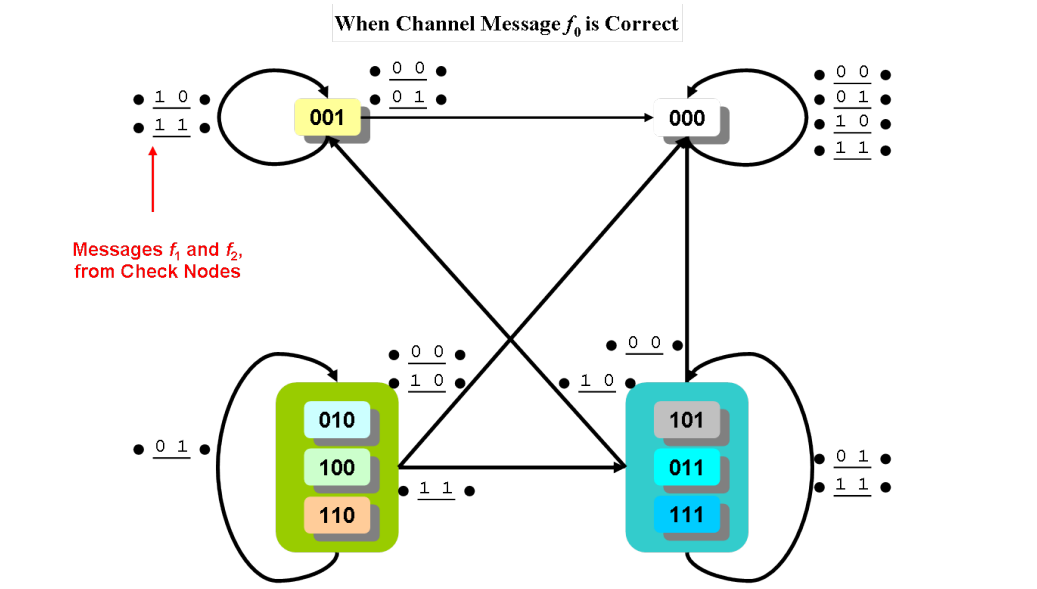


Figure 4.36: Memory state transition diagram when channel message is correct.

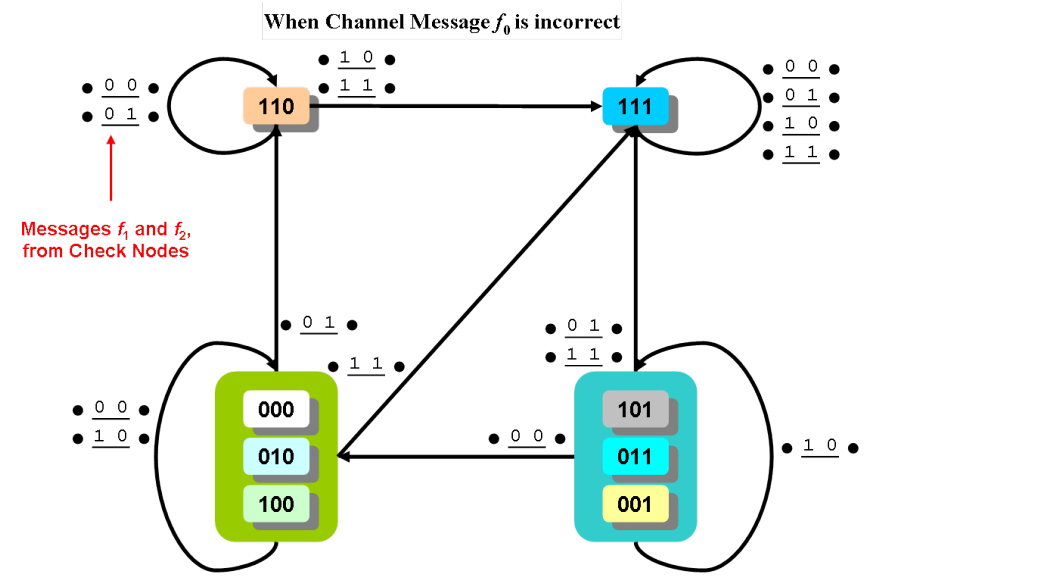


Figure 4.37: Memory state transition diagram when channel message is incorrect.

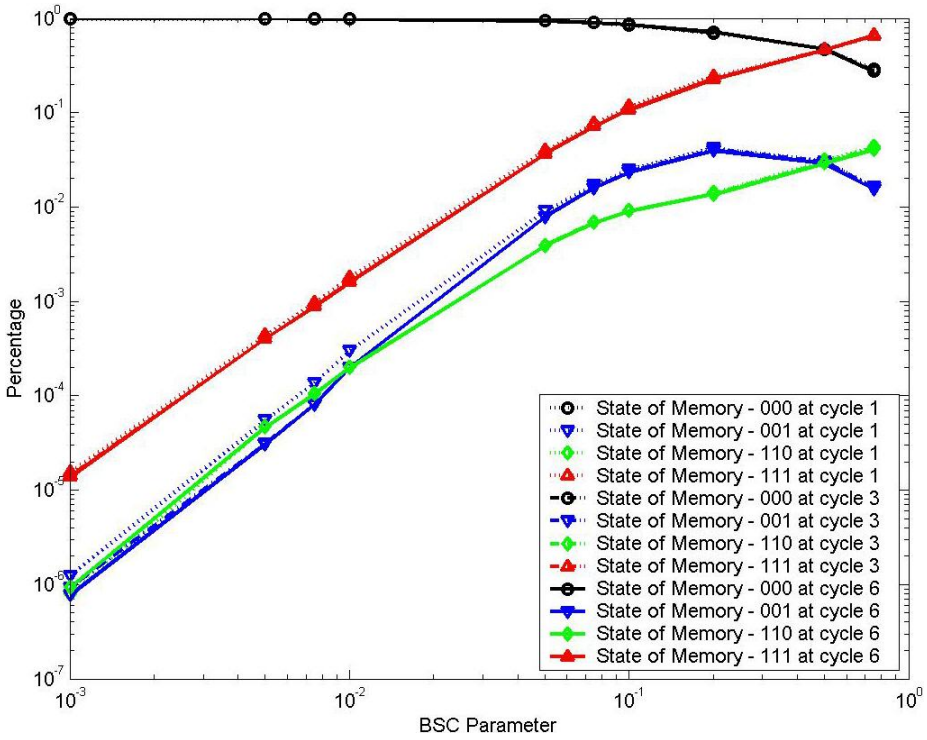


Figure 4.38: Monte Carlo Simulations on a (3,6) LDPC code of length 64 for memory state transitions, in the case of error-free decoding process.

(3,6) LDPC codes of length 64 are simulated for MCD while error-free decoding process is first considered. By varying BSC parameter ε, the state of C-element memory is transited with an increase of number iterations. The curves that indicate the proportion of memory state are plotted in Fig. 4.38. With the drawn results, when the channel parameter goes lower, namely the channel is less noisy, the decoded data is much more prone to be corrected then failed. For instance, when $\varepsilon = 0.001$, the MCD yields a gain of two orders of magnitudes in terms of error-correction. In addition, it is seen that the state of memory is barely transited by decoding iterations.

In the case of noisy decoding process, the MCD is simulated under an injection of internal fault of error-rate 0.0001. In Fig. 4.39, only the states of memory that generate an incorrect decoded data, i.e., “001”, “101”, “011”, and “111”, are considered. When BSC parameter flops to 0.001, the percentage of states of memory that induce an error in output is still lower than 0.0001. Therefore, even the transient fault rate within the decoder is as high as 0.0001, MCD still produces benefit in error-correction.

4.5.2.4 Roundup: Decoding Properties for MCD

Some decoding properties for MCD are round up by using trellis structure to analyze its decoding performance.

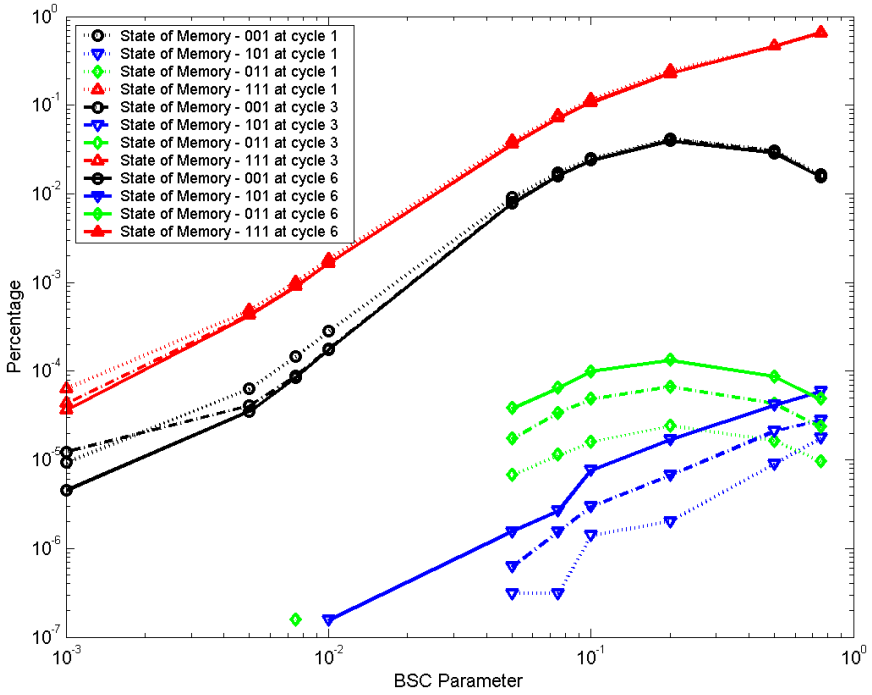


Figure 4.39: Monte Carlo simulations on a (3,6) LDPC code of length 64 for memory state transitions in the case of noisy decoding process. The internal error rate is 0.0001.

1. **Stability:** When the channel message is correct, the MCD easily ends up with a correct output. By contrast, if f_0 is incorrect, it turns out that the decoded data is contrarily stabilized to an incorrect output.

2. **Fast-convergence:** Since the state of memory is barely transited by decoding iterations, the MCD is able to perform error-correction even during a few number of iterations. For the noisy decoding case, more decoding iterations tends to enable better decoding performance.

3. **Resilience:** Even the internal fault rate is presented within the decoder, the MCD achieves efficient error-correction, namely an error-resilience decoder.

Thanks to the **Resilience**, a good decoder candidate for the ECC of cDMR is then introduced. Moreover, since the MCD hardly relies on decoding iterations from the **Fast-convergence**, a general power saving may be achieved by comparison with GBF in the case of noisy decoding.

4.5.3 Dynamic Power Saving With MCD

Considering the MCD implementation is on CMOS technology, the main power consumption can be decomposed of two contributions: static power and dynamic power [Sah91]. At first, static power is defined as the power used when the transistor is stable, namely when idle. However, some leakage dissipations diverts the sum of static power [Bak10] as follows: subthreshold current when idle, tunneling current through gate oxide, leakage current through reverse biased diodes, and contention current in ratioed circuit. However, these dissipations can be avoided. For instance, a special type of the CMOS transistor with near zero threshold voltage is the native transistor [BSU+99].

For the dynamic power, the total consumption includes the following aspects: charging and discharging of load capacitances [MKW08], short circuit power dissipation when the current can not flow through a direct path from V_{DD} to ground, and power glitches. The power glitches represents an unexpected value that is operated by the occurrence of unsteady-state inputs. In addition, the major dynamic consumption is still governed by capacitive charging/discharging.

CMOS power consumption Pow_{CMOS} can be expressed by the sum of static power Pow_{Static} and dynamic power $Pow_{Dynamic}$,

$$Pow_{CMOS} = Pow_{Static} + Pow_{Dynamic} \tag{4.12}$$

Furthermore,

$$Pow_{Static} = I_{static} V_{DD} \tag{4.13}$$

where I_{static} is the total current flowing on a device and V_{DD} is the supply voltage.

$$Pow_{Dynamic} = C_{load} V_{DD}^2 f_{clk} \tag{4.14}$$

where C_{load} is the capacitance and f_{clk} is the clock frequency. Since not each gate operates or switches at each clock cycle, (4.14) is often combined with factor α_f, so called the activity factor. Thus, the dynamic power dissipation may be expressed as

$$Pow_{Dynamic} = \alpha_f C_{load} V_{DD}^2 f_{clk} \tag{4.15}$$

With (4.15), it is found that operation/switch transition dissipates some energy as heat. The less operation/switch occurs, the less power consumes. To take into account the power consumptions of GBF and MCD algorithms, Fig. 4.40 shows two implementations for Gallager-A and MCD, respectively. By comparison to Gallager-A, MCD is able to reduce the dynamic power in the case of noisy decoding process thanks to its *Fast-convergence* and *Resilience* explained in Section. 4.5.2.4:

1. From *Fast-convergence*, MCD relies less on the decoding iteration number, thus, less transitions would be induced by increasing the number of iteration.

2. From *Resilience*, single transient upset occurs at one of the inputs of C-element can not result a temporary state to C-not block, the same for C-element's state memory. As pointed out in Fig. 4.40, unlikely Gallager-A architecture, C-not block usually retains in high impedance. In this case, glitches power dissipation is precluded.

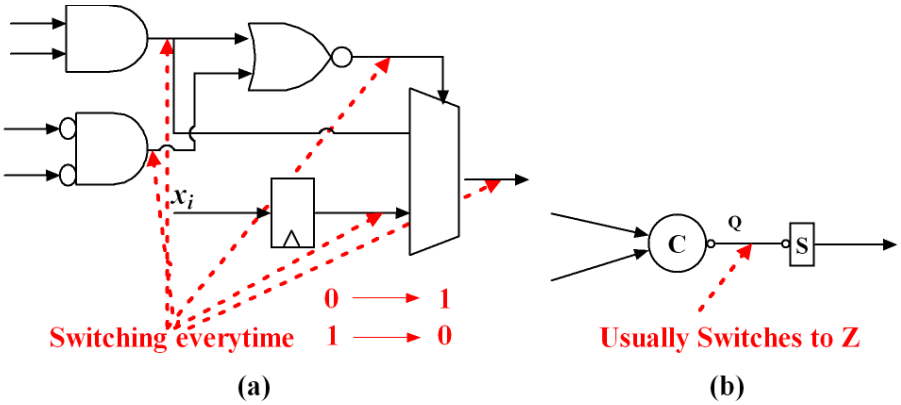


Figure 4.40: (a) Switching activity on a Gallager-A decoder architecture, (b) Switching activity on a MCD architecture.

4.5.4 Good Decoder Candidate for the ECC of cDMR

In this subsection, we demonstrate a good candidate for the ECC of cDMR. More details for cDMR can be found in Section. 4.3.4. By comparison to the GBF approach, the performance improvement achieved with MCD was analyzed thanks to two metrics: the BER performances over BSC under a error-free decoding process and faulty decoding process. For a sake of facility, only LDPC code with $d_v = 3$ is studied.

A (3,6) LDPC code of length 64 was simulated over BSC in the cases of error-free and noisy decoding process, respectively, as shown in Fig. 4.41 [TBW+13]. First, in the case of noisy decoding (dashed curves), the GBF performance worsens with increased iterations, but MCD does not suffer from this degradation. Significantly, when the length of a LDPC code is short, the MCD performs a gain in terms of decoding performance, as shown by the solid curves in Fig. 4.41. This short MCD decoder is a good candidate to the ECC block. Consequently, the cDMR technique based on the MCD architecture provides a general-purpose robust system to system design.

4.6 Conclusion

In this chapter, we first review some related fault-tolerant methods. A Muller C-element based fault-tolerant method for robust latch is first reviewed. An improved version of

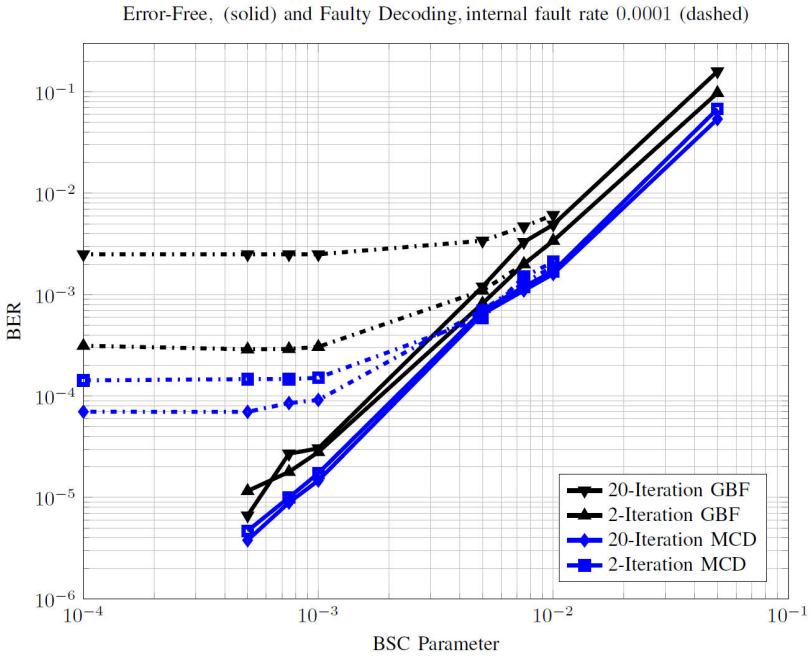


Figure 4.41: Simulation results of (3,6) LDPC code of length 64 under faulty and error-free decoding.

TMR based on Muller C-elements, as known as Restorative Feedback (RFB) [WLMT11], is able to increase the resilience against transient faults by two or three orders of magnitude compared to TMR.

After that, we present two solutions on the efficient error-correction in the presence of internal transient faults within decoder itself.

- **A Reliable Decoder Against Internal Transient Faults.** A decoding algorithm and its logic implementation is proposed for fast, low-complexity error correction in environments with a high rate of transient faults as well as hard errors. Thanks to the C-elements, the proposed method is also resilient against internal transient gate errors that may occur within the decoder itself.

- **A Space-Time Redundancy Technique.** A temporal majority logic element is applied at the decoder's output. It provides an additional dimension of redundancy. In fact, by applying this space-time redundancy technique, the decoder is able to ground its output error-probability below the internal transient error rate. Thus, the majority unit is supposed to be reliable.

Moreover, by examining the error-resilience from state memory of C-element, we study the capability of fault-tolerance inherited by C-element. Consequently, by com-

parison to a static logic gate, C-element is able to tolerates the upset events with error-resilience gain.

In addition, we investigate the properties of the reliable decoder against internal transient faults, referred as to MCD, thanks to density evolution theory [RU01] and trellis Structure [Ung82]. The properties of MCD can be summarized as follow:

1. **Stability:** When the channel message is correct the MCD easily ends up with a correct output. By contrast, if f_0 is incorrect, it turns out that the decoded data is contrarily stabilized at an incorrect output.

2. **Fast-convergence:** If the state of memory is barely transited by decoding iterations, the MCD is able to perform error-correction even during few a number of iteration. For the noisy decoding case, more decoding iterations tends to achieve better decoding performance.

3. **Resilience:** Even the internal fault rate is presented within the decoder, the MCD achieves efficient error-correction, namely the decoder against internal faults.

Furthermore, by comparison to Gallager-A technique, MCD algorithm may be able to reduce the dynamic power in the case of noisy decoding process.

Finally, we present a good decoder candidate for ECC of cDMR: a MCD of (3,6) short-length LDPC code. When the decoding process is error-free the MCD achieves also a gain in terms of decoding performance by comparison to the GBF, and produce error-correction benefit in a high error-rate of internal faults.

5 General Conclusion and Future Works

Electronic circuits have been entitled unreliable due to the affections of different faults. Faults experienced by semiconductor devices fall into three main categories: permanent, intermittent, and transient. Permanent faults are caused by manufacturing defects as variability from fabrications, or device wear-out [SH04]. They reflect irreversible physical degradations. Intermittent faults occur because of unstable or marginal hardware. They can be activated by environmental changes, for instance, higher or lower temperature and voltage [ENW08]. Many times, intermittent faults precede the occurrence of permanent faults. The transient defects and faults are becoming major impact to the reliability of electronic technologies. Due to the size downscaling processes, digital logic circuits are increasingly vulnerable to cross-talk, high-energy particle collisions and radiations [Bau01], thermal noise, electromagnetic interference, timing failures in synchronous pipelines, power supply noise, and other sources [KP74,ENW08,SH04]. Among these transient faults, Radiation-induced Soft-Error (SE) has led designers to too much attention on the topic of cosmic radiation induced transient faults in nanoscale computation. This event is called Single-Event Transient (SET) or Single-Event Upset (SEU) [Bau01].

Although materials could be fabricated as error-free theoretically with a huge cost for worst-case design methodologies [fS], the circuit is still susceptible to transient faults. On the contrary, much more costs can be reduced when the reliability issue is relaxed during the manufacturing process. In this case, an error-resilient design is imminently required for current logic systems or future nanoelectronics.

5.1 General Conclusion

Classically, all the processes in electronic digital circuits can be classified into three main categories: storage units, such as the memories; communication units, such as the buses; and computing units, such as the logic gates or adder. The scheme of an operation process without applying ECC strategies can be described as the non-shaded parts of Fig. 5.1. The result g is obtained from an operation $F(x)$ by given the data x as input.

In this thesis work, we put our emphasis on the application of ECC on unreliable

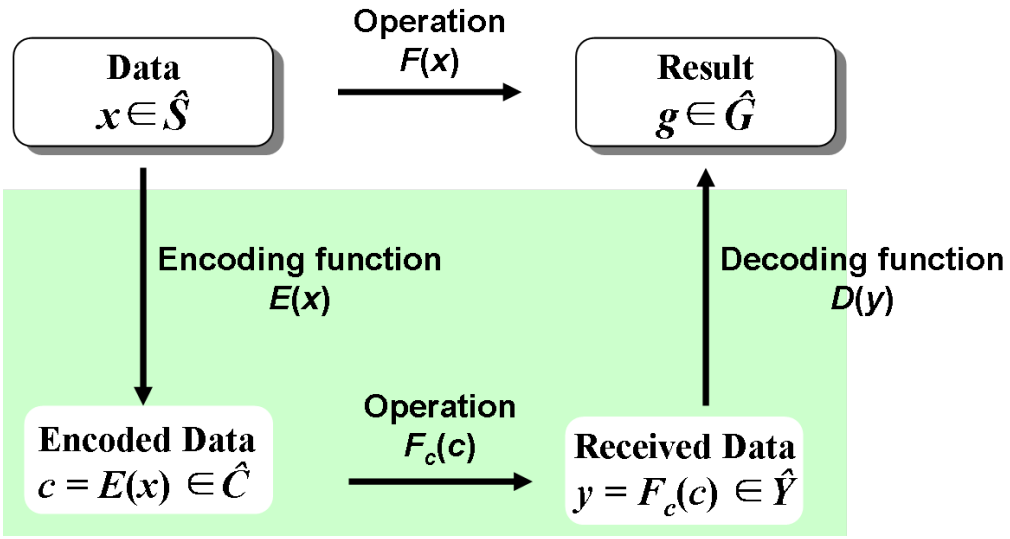


Figure 5.1: A formal model for the application of ECC on unreliable circuit. The non-shaded parts are the scheme of an operation process without applying the ECC strategy.

circuit. A formal model for the application of ECC on unreliable circuit is illustrated as the shaded parts in Fig. 1. The model consists of three blocks: the encoding function $E(x)$ to produce a codeword $c \in \hat{C}$, the computing function $F_c(c)$ to perform the operation and the decoding function to recover the result g from an error-correction process applied to the received data y. Please note that all the three blocks are vulnerable to transient faults and permanent defects. As a result, in order for the ECC block to recover the result from the output of computing function $F_c(c)$, the output of function $F_c(c)$, y, has to be a linear code. To achieve this end, there are four cases. We introduce the four cases as the contributions of this thesis work presented as follow.

5.1.1 The Main Results

In this part, we summarize the main results of this thesis work. In short, we first expounded the current major concern: electronic circuit is trending unreliable. The related fault-tolerant methods to deal with the addressed problem were reviewed. The contributions of this work were presented as the order of the outlines as follows:

To obtain a linear code y, the first case is that a group homomorphism is obtained among $E(x)$ and $F_c(c)$. An error-correction example of this solution is presented in Chapter 1. More precisely, we introduce a fast arithmetical fault-tolerant method based on residue number system (RNS) [Gar59] for more robust computations. The RNS provides a very fast arithmetic due to its capability of performing the carry-free operations, i.e. addition, subtraction and multiplication. By adding some redundant residues, the RNS has an error detection and correction property that is called Redundant Residue Number System (RRNS) [ST67]. However, the data restoration of a conventional RRNS

is an iterative decoding process. In this case, it requires high latency to achieve an error-correction. In this chapter, we propose a technique, referred as to bidirectional redundant residue number system (BRRNS) [TBJJ10], that achieves fast arithmetic and enables to mask an overall SET. In this method, the detection and the error-location are simultaneously performed in a plural parallel consistent-checking that has the capability of locating the corrupt digit. Therefore, a fast error-correction is obtained. An efficient pipeline architecture for the BRRNS based ripple-carry adder is illustrated as well.

Secondly, when the received data y is obtained from the repetitions of computation $F(x)$, it is a linear code as well. Such like, the Triple Modular Redundancy (TMR) method [LV62], its encoding function is obtained by two duplications of the original computing function. TMR has been employed as the fault-tolerant technique for a robust system almost since fifty years ago. However, is it efficient to apply triple modular redundancy technique to increase the robustness of an unreliable computation? Additionally, how can we make a fair comparison between two fault-tolerant methods?

Chapter 2 presents a novel criteria, called Reliability-Efficiency Criteria (RE-Criteria) [TBJJ11], defined by two dimensional space of architectural solution: error probability at fanout and hardware efficiency estimations. In this system, taking into account that the correction unit is also subject to error, we propose to assess the quality of an architecture by using not only its efficiency (i.e. the normalized number of operation per area and per unit of time), but also its output error rate. Thanks to this dual space criteria, designers are able to discern the properties, the reliability and the efficiency, for different fault-tolerant strategies.

The third case is that the processes in the circuit are storage or communication. Thus, the ECC strategy can be directly applied. A solution to the subject of embedded binary error-correction is first presented in Chapter 3.

A Fast Decoding Method for Binary Symmetric Channel: A modified version of Gallager Bit-Flipping decoding method (GBF) has been proposed for decoding Low-Density Parity-Check (LDPC) codes over a Binary Symmetric Channel (BSC) in [CV09]. The modification is about the estimate of a variable node that is taken to be the majority of the incoming messages at the end of each iteration. In this work, we propose another similar modification. Via density evolution analysis, the majority-vote modification enables to increase the decoding process. Since slight extra hardware resource overhead is required, our method is appropriate for fast low-power error-correction design.

The fourth case is to directly generate the data y for the decoding function, by a concatenation of two outputs from the original function and a parity-mapped function. This model, called coded Dual-Modular Redundancy (cDMR), is based on the Dual-Modular Redundancy (DMR). Note that a parity-mapped function replaces a duplicated original function. The cDMR is a modified version of an error-correction model, referred as to Low-Density Parity-Check (LDPC)-coded Fault Compensation Technique (LFCT), that was proposed by Winstead in [WH09].

An Embedded Error-Correction Model: The proposed robust model, cDMR, is shown in Fig. 5.2. The concept of cDMR is using an ECC block to confront with the transient faults that occur in the main function block and within ECC block as well. Thus, it requires an ECC block that is able to resist its own internal transient errors. Otherwise, the whole cDMR system can not perform an efficient error-correction under a faulty decoding process. Moreover, the extra overhead function, parity-mapped function $H \cdot F(u)$ generates the redundancy bits according to the output of the main function $F(u)$. Please note that, in practice, it needs that $F(u)$ and $H \cdot F(u)$ are built in a special fashion to precluded the correlated at their outputs. The most reliable approach is to use a flat truth-table synthesis, as is done with cross-bar logic arrays [ROK09]. This approach guarantees that error events occur independently on all the codeword bits, so that correlated error bursts are precluded.

This embedded error-correction model, cDMR, can be adapted into any unreliable design whenever two issues are avoided: correlated errors in original function's output and also the parity-mapped function's output, and the ECC circuit is able to perform efficient error-correction in the presence of internal transient faults. Thanks to the cross-bar technique, the correlated-error issue can be avoided in cDMR. In Chapter 4, we focus on an efficient error-correction in the presence of internal transient faults within decoder itself.

A Decoder Against Internal Transient Faults: Muller C-element based Decoder (MCD) is a decoding technique that is able to perform error-correction in the presence of hight rate of internal transient faults. The MCD is an improved version of GBF, based on the Muller C-elements to carry out the variable node process. In practice, for the decoding process of MCD, the messages pass through variable nodes to check nodes are more reliable. By comparison to the GBF, when the internal error rate is as high as 0.01, the MCD does not exhibit a Bit Error Rate (BER) performance degradation thanks to an increase of the number of decoding iterations. Therefore, the MCD can be considered as a decoder who inherits the potential of error-resilience to transient upsets of Muller C-elements.

A Space-Time Redundancy Technique to Improve the Decoding Performance: To further increase the BER performance of a decoder in the presence of internal faults, the Space-Time redundancy technique can be applied. In fact, a majority voter at the end of decoding step enables to sample each output during the last partial decoding iterations. Then it outputs the decision from a majority of sampled data. Thanks to the Space-Time technique, the transients that induced flipped over at the decoded output is able to be masked by the correct outputs. In addition, the proposed technique can be widely adapted for any message-passing decoder to further improve the BER performance in the presence of internal errors.

Moreover, by examining the error-resilience from state memory of C-element, we study the capability of fault-tolerance inherited by C-element. Finally, we present a good decoder candidate for ECC of cDMR: a MCD of (3,6) short-length LDPC code. When the decoding process is error-free the MCD achieves also a notable gain in terms

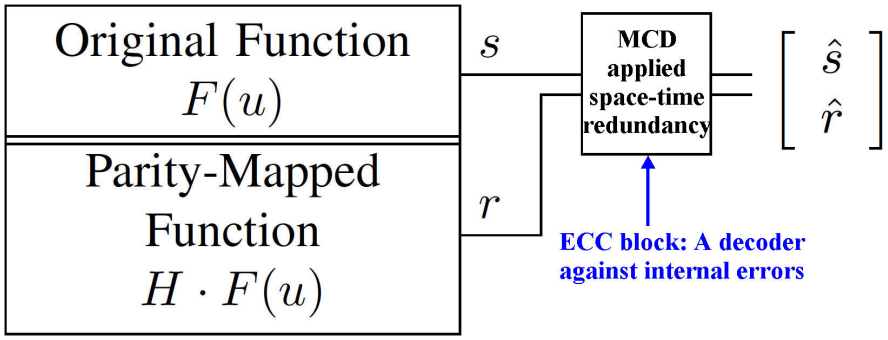


Figure 5.2: The cDMR architecture based on the MCD. The space-time redundancy is applied to the MCD (as an ECC block).

of decoding performance by comparison with the GBF. Moreover, it introduces error-correction benefit in a high error-rate of internal faults.

5.2 Future Works

In this section, some guidelines for fault-tolerant design are first proposed. Then, we discuss some perspectives and future works for this study.

5.2.1 Guideline for fault-tolerant design

Researchers have been putting emphasis on hardware complexity, power consumption, delay overhead, and etc., for a good robust design on unreliable circuit. However, it is quite complicated to design an efficient circuit, which achieve high performance and high reliability with a low power consumption. In this case, a curve drawn in Fig. 2.5 that points out two criteria (robustness and energy efficiency) allows designers to realize a good fault-tolerant design.

In order to approximate the curve, low power and high reliability, we advocate two guidelines in this work:

1. **Inherence**: the function or restoring block inherits the potential of error-resilience.

2. **Tradeoff**: by decreasing some robustness, an energy efficiency hardened design is achievable, vice-versa.

For the first point, a good example is the MCD method. The MCD method employs Muller C-elements and XORs to realize a error-resilient message-passing decoder. The C-element can be considered as a component that inherits the potential of error-resilience. Such that, the MCD is beneficial as resilient to errors from the hardware architecture itself.

For the **Tradeoff**, a good example is the general-purpose robust model, cDMR. Even though the cDMR may burden the overhead of design to preclude the correlated errors from main function $F(u)$ and parity-mapped function $H \cdot F(u)$, it significantly improves system's robustness. More precisely, if we use cross-bar approach to design the $F(u)$ and $H \cdot F(u)$, the cross-bar itself already is a fault-tolerant circuit to manufacturing defects/variability or transient/intermittent faults. Consequently, cDMR model may induce more hardware complexity compared to the original design, but huge gain in terms of reliability can be obtained.

5.2.2 Perspectives and Directions for future work

At first, according to the BER results obtained in Fig. 4.41, the MCD of a short length of LDPC code is compatible with data words in current 32-bit processors. Furthermore, some modified MCD techniques would be also helpful to achieve some featured decoder, such like, low hardware complexity or low delay. In addition, according to the presented BER results, the GBF method adapted space-time redundancy allows an efficient error-correction under a noisy decoding process. As a result, the decoding methods proposed in this work can be widely applied into contemporary logic systems and as well as future nanoelectronic architectures for the purpose of robust design.

Moreover, some featured tolerant techniques can also be adapted in different designs. Such like, the BRRNS, see Section. 1.3, a fast arithmetical fault-tolerant method based on residue number system facilitates the computation via carry-free operations. In the presence of internal transient upsets, the BRRNS is able to correct a single SET without adding any delay to the design. Likewise, thanks to a majority voter at the end of decoding process, the space-time technique can significantly improves the decoder's BER performance while internal transient gate errors that occur within the decoder itself. The space-time technique is suitable for a low redundancy robust decoding design.

To take into account the low-level analysis results, such like, power, delay, hardware complexity, and etc., the designers have to go through the circuit design process. Thus, the designers have to carry out each circuit design flows independently for different architectures. In this case, enormous time and research resources are wasted. By using the RE-criteria, the designer is able to easily and quickly analyze different strategies and methods, at architectural level, thanks to estimation results in terms of hardware efficiency and robustness.

In order for further revealing the benefit of the cDMR architecture implemented the MCD applied space-time redundancy (Fig. 5.2), a typical case is necessary. Please note that a further discussion is needed before the implementation of the cDMR architecture. Some issues have to be studied. Such like, the latency made by the ECC block and the extra hardware complexity to carry out a cross-bar style computation function and a cross-bar style parity-mapped function as well. Consequently, the cDMR architecture may reveal a whole new view for designing robust architectures. Researchers may also be inspired from the proposed technique. But, a corresponding analysis in future is

necessary for the amelioration.

At last, as a direction for future work, we focus on the multi-layer fault-tolerant design. For instance, the adder is to be used in a pipeline of arithmetic operations. The initial inputs to the adder are reliable. It means that all errors originate from within the circuit. The adder is to be cascaded with other adders in the pipeline mode, so that we need to mask errors before delivering data to the next stage. If the cDMR model is applied into the pipelined adder design, it will be multi-layered cDMR model. The only left task is to figure out the ECC block of cDMR to cope with the transient faults in pipeline.

A Annex

A.1 Towards to the SET

In 1975, Binder *et al.* [BSH75] firstly addressed a confirmed report of cosmic-ray-induced upsets in space, where four upsets in 17 years of satellite operation were observed in bipolar J-K flip-flop operating. 26 years later, Baumann [Bau01] distinguished the "soft" error from the "hard" error. He defined a "soft" error as a type of event which occurs in semiconductor devices by two dominant radiations: alpha particles and cosmic radiation. This event is called Single-Event Transient (SET) or Single-Event Upset (SEU). Accordingly, Soft Error (SE) is a transient fault induced by radiation strike, where the circuit itself is undamaged. In contrast, the un-restorable fault is called Hard Error (HE). Although SET depends on the mass, energy, charge of the particle, device, etc. [Mes82], Benedetto *et al.* [BEM+06] measured the SET pulse width by giving Linear Energy Transfer (LET), which is the factor observing the impact of highly energetic particles. Since the SET width density per circuit node follows a normal distribution, the SE simulator relying on the computer programs [Sri96] has been developed for facilitating the observation and the mitigation of SE.

A.2 A Quick View into Error-Correction

The very first Error Correction Code (ECC) has been invented by *R. Hamming* in 1950 [HAM50], when he intended to prevent the soft system hitch from PC. Since then, the ECC has been rapidly and drastically developed. For instance, two famous error control codes, LDPC code and Turbo Code [Gal63, Moo05], are currently utilized in modern telecommunication applications. Likewise, the message-passing decoding method is as well one of ECC family.

Here comes a story which reveals the essence of Error-Correction, from which you may have first view of Error-Correction:

One day, (story goes always like this), A would like to know whether the sky is blue, from his own point view, it is. In order to confirm this fact, he needed to ask others' observations to the question. Then, B said, the sky is not blue. As such, A was looking for the observation from the 3rd, C thus stood up to give his thought, which is, the sky

is not blue. At last, A figured out the sky is not blue...

Error Detection

In this case, if A and B disagrees at a point, then a disagreement is presented, that's so called the occurrence of an error. As a technical term, it's the error detection or error checking, of which B is a redundancy to A (the source). As explained in Hamming's theory [HAM50], to detect an error (in the presence of single error), an additional word/bit (for our story is B). Otherwise, A's observation is correct when A and B agree.

Like wise, extending this principle into multiple-ary case, A_1, A_2, A_3,... A_k are looking for confirmations, then B_1, B_2, B_3... B_m are needed to check As' values.

Error Correction

In general, the decoding process comprises error-detection and error-correction. To the story, the observations from B and C are supposed to be the decoding process, where B's observation is the error-detection. In theoretical, at least two obverses (B and C) are required to perform the correct a single error (if A is erroneous, B and C revise A's thought.) Similarly, in the case of A is multi-ary, A_1, A_2, A_3,... A_k, B_1, B_2, B_3... B_m are involved. To be precise, in the modern error-correction decoder, the number of redundancy (number of Bs) are flexible due to the requirement of the system and the overcharge by decoder per-se.

Belief Propagation (well known decoding algorithm used in modern telecommunication system, such like the modified version for LDPC codes [KFL01])

To continue the previous story, $'\rightarrow'$ represents the message circulating,
$A \rightarrow B \rightarrow C \rightarrow A \rightarrow B \rightarrow C \rightarrow A$ .... if more observes are involved, then, $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow$ ..... $\rightarrow A \rightarrow B \rightarrow C$ In a short, the message circulating, or message-passing based decoding processing is the essential of Belief Propagation by Pearl. More precisely, each node in the observation circle (variable nodes and parity check nodes are assembled a codeword), send over a observations to his neighbors (the connections are given from encoding matrix, which generates the redundancy from source bits). Once the iterative message-passing process is done (stopped neither by giving iteration numbers or a syndrome procedure, which means a procedure check all the observes match certain criterion.) A noble work developed by *Kschischang*, Sum-Product Algorithm [KFL01], considers all the algorithms as two sets of functions connected by bipartite graph (factor graph) to perform internal computations/message updating. The belief propagation can be derived from sum-product algorithm as well.

A.3 A Basic Co-Design

In this section, some experiences to the topic "Co-Design: Soft and Hard" are contributed to the hardware architecture design for electronic device in different domains, such like, FFT in Digital Signal Processing (DSP), Modern Communication Coding IP, RNS, and etc.

For last two decades, the circuit design procedure has been changed a lot due to the rapid development of chip manufacturing. It is in view that this kind impact would be more intense. Moreover, many EDA tools are emerging to facilitate the design to the question, e.g., Cadence features the block 'Mixed Signal Analysis' to cop with the hybrid of Analog & Digital. Even though, a basic idea to proceed the circuit design would be useful and practical, which is explained as following:

- *User Customized*

  The very begin of a design always goes with the specification. To this aspect, a specification is supposed to be detailed as much as it'd be needed. Such as, the general description, the behavior in function, the requirements, and etc.

- *Pre-Verification by 'Soft'*

  Right after, a verification for the system's behavior is necessary, whereas the algorithms, methods, and the system level control are verified. In addition, a test-bench needs to be drawn to the system. In general, this top-down kind approach, it's so called, Pyramid hierarchical approach.

  To be precise, according to the specification, the systematic description is held at the top, then several main functions are spotted afterwards, for which a controller that wraps them up is required. Subsequently, concerned son-functions or methods and algorithms are applied in a topology way.

  As such, C/C++, 'software' description languages, would be fit to accomplish this task. BTW, for the current tools, basic compilers or MatLab, and etc.

- *Pre-Fabrication by 'Hard'*

  In this task, an inverse Pyramid hierarchical approach performs the same kind construction to the 'soft' verification, instead of using 'soft' description language, but Hardware Description Language (HDL). At first, the son-functions or algorithms and methods are written in HDL, then, step by step, till the bottom of the Pyramid. Once the whole system is established, a test-bench is needed to simulate design's behaviors.

  For example, the well-known Monte Carlo Simulations, input are fed into the design then compare the output to the corresponding ones. In this case, the tool,

ModelSim, would be competent for it. Moreover, some HDLs are most used, such as VHDL, Verilog HDL, System C, System VHDL, System Verilog HDL, Verilog-AMS (featured for analog design or mixed design).

Towards the System C's usage, it can be used as a HDL or as a Software Modeling Language (SML). It is a hybrid platform that spans the gap between these design domains. For abstract modeling of signal-processing architectures, System C is, much more flexible and offers more descriptive power than System Verilog. It lacks some of the low-level modeling abilities of Verilog-AMS (e.g. it cannot handle continuous-time nonlinear device models), but it can model discrete-time analog operations via real-value modeling (RVM). System C RVM is more flexible than System Verilog or Verilog-AMS RVM.

- *Co-Simulation*

  Since the Pre-Fabrication has been done, the design is 'ready' to be implemented when the test-bench shows that your design matches the specified behaviors, as the 'soft' verification as well. For sure, some tools support the Co-Simulation property, for example, a synthesis tool called GAUT is able to transfer SML to HDL, vice-versa. For that sake, researchers can largely reduce the developing period.

  Once again, the key point to this step is that make sure your SML simulations and HDL simulations are total matched.

- *Circuit-Level verification*

  Two approaches may lead the directions: FPGA and ASIC/ASIP. The former is suitable for short term development since FPGA is proposed for feasible circuit design, where the design can be synthesized as equivalent unit respecting to transistors in ASIC. *Xilinx* and *Altera* are both most used FPGA families, those kit accompanied with customer develop tools are compatible for many kinds circuit design. Moreover, for the DSP, T&X Instrument may the best choice.

  If the design is made to be implemented into circuit, in this case, the wafer production requires ASIC sort verification. Cadence & Synopsis, those two powerful tools provides from high-level to low-level synthesis and verification as well. For example, to the digital kind, generating Netlist by HDL then forward it to the design test, the post-implementation test.

- *Confirmation before fabrication*

  After all these, your design may ready to be implemented. However, it's been always complicated to prepare a design to be fabricated due to the hardware

resource is often over charged, such like the power consumption, the overall cadence, and the circuit's reliability, and etc. Sometimes, the circuit design backwards to the previous step to ameliorate its performances and assure the design is implementable.

A.4 Programming on Matlab for Pareto Curves

In this section, we explain how the Pareto curves shown in Fig. 2.6 (see Section. 2.4.2) are drawn. Here, we illustrate the programming on Matlab for generating the curves of the S-TMR.

Before we expatiate the programming, some points should be remarked, such as followings:

- Four FIR filters that are varied by order of filter and also bit length of input are referred in Fig. 2.6. In addition, each FIR filters are designed in following strategies: the simplex (as the reference), S-TMR, BRRNS based (as the ECC solution), ARQ via Modulo 4, ARQ via Modulo 8, ARQ via Modulo 16, and ARQ via spatial DMR.

- From the equations (2.3), (2.4), and (2.5), in order to calculate the (P,γ) the number of area unit and the number of time unit are needed. In this work, we use the results obtained at FPGA as the analysis data, as shown in Table. 2.1. More precisely, the number of Slices occupied is considered as the number of area unit. Moreover, the number of time unit is equal to one due to the pipeline design.

- Without loss of generality, the number of area unit is normalized in our programming (the number of time unit still stays the same as one). In fact, the number of area unit for each solutions are normalized by the corresponding number of area unit of the simplex. For instance, from Table. 2.1, for FIR-I (16,5) as (order of filter, bit length of input), the number of area unit of the simplex is 94, and the number of area unit of ARQ-DMR is 207. To calculate $P_{ARQ-DMR}$ and $\gamma_{ARQ-DMR}$, the number of area unit n for ARQ-DMR is equal to $207/94$.

- At last, the error probability in decibel shown in axes-Y in Fig. 2.6 is obtained by evaluating ten times the base-10 logarithm of the error probability to p_e.

The programming to produce Pareto curves for S-TMR is shown below:

—

1: The Simplex, the original function which has no redundant involved.

—

– error probability of unitary area,
– it's set to 10^{-13}

$P_e = 10^{-13}$;
– *number of area unit*
– *as the oder of (FIR-I FIR-II FIR-III FIR-IV)*
$n_{simplex}$ = *[94 128 171 232];*
– *number of time unit*
– *Pipeline Designing*
$m_{simplex}$ = *[1 1 1 1];*
—
2: The Spatial-TMR (S-TMR) based
—
– *The Voter in the S-TMR*
– *number of area unit*
– *as the oder of (FIR-I FIR-II FIR-III FIR-IV)*
n_Q = *[19 19 28 36];*
– *number of area unit*
– *as the oder of (FIR-I FIR-II FIR-III FIR-IV)*
$n_{stmr} = n_{simplex} \cdot 3 + n_Q$
– *number of time unit*
– *Pipeline Designing*
m_{stmr} = *[1 1 1 1];*
– *Normalizations*
for i = *1:4*
$n_{stmr}(i) = n_{stmr}(i)/n_{simplex}(i)$;
$n_Q(i) = n_Q(i)/n_{simplex}(i)$;
end;
– *the S-TMR,* P_{S-TMR} *and* γ_{S-TMR}
for i= *1:4*
– *error probability from the computing function*
$P_R(i) = (1 - (1 - P_e)^{n_{simplex}(i)})$;
– *error probability from the voter*
$Q(i) = (1 - (1 - P_e)^{n_Q(i)})$;
$P_{S-TMR}(i) = 3 \cdot (P_R(i))^2 \cdot (1 - P_R(i)) + P_R(i)^3 + Q(i) \cdot (1 - (3 \cdot (P_R(i))^2 \cdot (1 - P_R(i)) + P_R(i)^3))$;
$P_{S-TMR}(i) = 10 \cdot log10(P_{S-TMR}(i)/p_e)$;
$\gamma_{S-TMR}(i) = n_{stmr}(i)$;
end;

A.5 Software for Low Density Parity Check (LDPC) codes

All the simulated LDPC codes are generated by using the *alist from Software for LDPC codes (*Release. LDPC-2006-02-08) in [Nea]. An *alist contains the following informations: size of the matrix, list of integer coordinates where the non-zero entries are, weight of each row, and weight of each column. For example, the* alist of $N = 64, M = 32$

32 64 – M, N
6 3 – d_c, d_v
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 – weight of each column

*3 3 3

1 17 18
1 22 29
7 26 27
7 14 31
15 17 18
10 22 29
9 19 21
12 16 22
3 6 7
8 11 19
2 13 16
13 18 26
12 21 31
1 3 24
4 21 28
5 19 21
4 6 14
16 26 32
3 10 27
9 15 28
3 9 25
6 8 23
4 23 31
14 15 20
9 12 27
16 23 25
6 18 28
5 15 20
8 11 24
20 25 28
2 14 32
1 2 13
19 25 27
26 27 30
3 5 30
2 3 4
9 16 23
11 26 30

11 17 28

5 9 31

19 20 22

15 18 26

5 17 30

6 7 32

8 15 30

7 8 27

23 24 32

7 10 16

1 4 20

12 25 30

6 10 13

1 8 22

5 24 31

11 23 31

14 18 29

13 20 21

13 29 32

4 10 29

17 24 28

2 14 24

11 12 19

A.6 Acronyms

ANT	Algorithmic Noise-Tolerance
ARQ	Automatic Repeat reQuest
ASET	Algorithmic Soft-Error Tolerance
BER	Bit Error Rate
BEX	Base Extension
BiAWGN	Bi-Additive White Gaussian Noise
BISER	Built-In Soft Error Resilience
BRRNS	Bi-directional Redundant Residue Number System
BSC	Binary Symmetric Channel
cDMR	coded Dual Modular Redundancy
CRT	Chinese Remainder Theorem
DMR	Double Module Redundancy
DSP	Digital Signal Processing
ECC	Error Correcting Code
ERSA	Error Resilient System Architecture
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FSM	Finite State Machine
GBF	Gallager's Bit-Flipping (decoder)
LDPC	Low-Density Parity-Check code
LFCT	LDPC-coded Fault Compensation Technique
LSB	Least Significant Bit
MCD	Muller C-element based decoding
MCDfb	Muller C-element based decoding with feedback mechanism
MGBF	Modified Gallager's Bit-Flipping (decoder)
ML	Maximum Likelihood
MPD	Message-Passing Decoding
MRC	Mixed Radix Conversion
MSB	Most Significant Bit
PSF	Probability Signal Flow
RE-Criteria	Reliability-Efficiency Criteria
RFB	Restorative Feedback
RNS	Residue Number System
RPR	Reduced Precision Redundancy
RRNS	Redundant Residue Number System
S-TMR	Spatial-TMR
SE	Soft-Error
SET	Single-Event Transient
SER	Soft-Error Rate
SEU	Single-Event Upset
SOI	Silicon-On-Insulator(technology)
T-TMR	temporal-TMR
TMR	Triple Modular Redundancy
VOS	Voltage Overscaling

A.7 Publications

- International Conference

    1. *Y. Tang*, E. Boutillon, C. Winstead, C. Jégo, and M. Jézéquel, "Muller C-element based Decoder (MCD): A Decoder Against Transient Faults," in *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, may 2013.
    2. C. Winstead, *Y. Tang*, E. Boutillon, C. Jégo, and M. Jézéquel, "A Space-Time Redundancy Technique for Embedded Stochastic Error Correction," *Turbo Codes and Related Topics, 2012 7th International Symposium on*, august 2012.
    3. *Y. Tang*, C. Winstead, E. Boutillon, C. Jégo, and M. Jézéquel, "An ldpc decoding method for fault-tolerant digital logic," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, may 2012, pp. 3025-3028.
    4. *Y. Tang*, E. Boutillon, C. Jégo, and M. Jézéquel, "Hardware efficiency versus error probability in unreliable computation," Signal Processing Systems (SiPS), 2011 IEEE Workshop on, October, 2011, pp. 168 -173.
    5. *Y. Tang*, E. Boutillon, C. Jégo, and M. Jézéquel, "A new single-error correction scheme based on self-diagnosis residue number arithmetic," *Proc. Design and Architectures for Signal and Image Processing (DASIP 2010)*, Edinburgh, Scotland, 2010.
    6. E. Boutillon, *Y. Tang*, C. Marchand and P. Bomel, "Hardware Discrete Channel Emulator," *Pro. High Performance Computing and Simulation (HPCS 2010)*, pp. 452 - 458, Caen, France, 2010.

- International Workshop

    1. *Y. Tang*, G. Sundararajan, C. Winstead, E. Boutillon, C. Jégo, and M. Jézéquel, "Techniques and prospects for fault-tolerance in post-CMOS ULSI," *21st International Workshop on Post-Binary ULSI Systems, ULSIWS*, Victoria, Canada, may, 2012.

References

[AKS93] V.D. Agrawal, C.R. Kime, and K.K. Saluja. A tutorial on built-in self-test. i. principles. *Design Test of Computers, IEEE*, 10(1):73 –82, mar 1993. 35

[AM08] S. Almukhaizim and Y. Makris. Soft error mitigation through selective addition of functionally redundant wires. *Reliability, IEEE Transactions on*, 57(1):23 –31, march 2008. 6

[ARB+07] J-L. Autran, P. Roche, J. Borel, C. Sudre, K. Castellani-Coulie, D. Munteanu, T. Parrassin, G. Gasiot, and J-P. Schoellkopf. Altitude see test european platform (astep) and first results in cmos 130 nm sram. *Nuclear Science, IEEE Transactions on*, 54(4):1002 –1009, aug. 2007. 36

[Bak10] R. J. Baker. *CMOS: Circuit Design, Layout, and Simulation, Third Edition*. Wiley-IEEE, 2010. 90, 109

[BAS05] S. Baloch, T. Arslan, and A. Stoica. Efficient error correcting codes for on-chip dram applications for space missions. In *Aerospace Conference, 2005 IEEE*, pages 1 –9, march 2005. 5

[Bau01] R. C. Baumann. Soft errors in advanced semiconductor devices-part i: the three radiation sources. *IEEE Transactions on Device and Materials Reliability*, 1(1):17–22, 2001. 5, 37, 113, 121

[BEM+06] J. M. Benedetto, P. H. Eaton, D. G. Mavis, M. Gadlage, and T. Turflinger. Digital single event transient trends with technology node scaling. *Nuclear Science, IEEE Transactions on*, 53(6):3462 –3465, dec. 2006. 121

[BJ84] Jr. Bell, M. and W. Jenkins. A residue to mixed radix converter and error checker for a five-moduli residue number system. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '84.*, volume 9, pages 242 – 245, mar 1984. 25, 28

[BM73] F. Barsi and P. Maestrini. Error correcting properties of redundant residue number systems. *Computers, IEEE Transactions on*, C-22(3):307 – 315, march 1973. 21, 24, 25

[BM74] F. Barsi and P. Maestrini. Error detection and correction by product codes in residue number systems. *Computers, IEEE Transactions on*, C-23(9):915 – 924, sept. 1974. 24, 25

[BRU04] Louay Bazzi, Thomas J. Richardson, and Rüdiger L. Urbanke. Exact thresholds and optimal codes for the binary-symmetric channel and gallager's decoding algorithm a. *IEEE Transactions on Information Theory*, 50(9):2010–2021, 2004. 58

[BSH75] D. Binder, E. C. Smith, and A. B. Holman. Satellite anomalies from galactic cosmic rays. *Nuclear Science, IEEE Transactions on*, 22(6):2675 –2680, dec. 1975. 121

[BSU+99] H. Banba, H. Shiga, A. Umezawa, T. Miyaba, T. Tanzawa, S. Atsumi, and K. Sakui. A cmos bandgap reference circuit with sub-1-v operation. *Solid-State Circuits, IEEE Journal of*, 34(5):670 –674, may 1999. 109

[BTMB10] Emmanuel Boutillon, Yangyang Tang, Cedric Marchand, and Pierre Bomel. Hardware discrete channel emulator. In *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, pages 452 –458, 28 2010-july 2 2010. 57

[Che62] P. W. Cheney. *An investigation of residue number theory for digital systems*. Ph. D. dissertation, Stanford Univ., Calif., 1962. 20

[Con08] R. Conway. Efficient residue arithmetic based parallel fixed coefficient fir filters. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pages 1484 –1487, may 2008. 45

[CP93] H. Cha and J.H. Patel. A logic-level model for alpha;-particle hits in cmos circuits. In *Computer Design: VLSI in Computers and Processors, 1993. ICCD '93. Proceedings., 1993 IEEE International Conference on*, pages 538 –542, oct 1993. 6

[CRA06] J. Chang, G.A. Reis, and D.I. August. Automatic instruction-level software-only recovery. In *Dependable Systems and Networks, 2006. DSN 2006. International Conference on*, pages 83 –92, june 2006. 5

[CV09] S.K. Chilappagari and B. Vasic. Error-correction capability of column-weight-three ldpc codes. *Information Theory, IEEE Transactions on*, 55(5):2055 –2061, may 2009. 11, 51, 52, 58, 67, 115

[DL95] P. Dahlgren and P. Liden. A switch-level algorithm for simulation of transients in combinational logic. In *Fault-Tolerant Computing, 1995. FTCS-25. Digest of Papers., Twenty-Fifth International Symposium on*, pages 207 –216, jun 1995. 6

[DL05] A. DeHon and K.K. Likharev. Hybrid cmos/nanoelectronic digital circuits: devices, architectures, and design automation. In *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, pages 375 – 382, nov. 2005. 86

[EJ80] M. Etzel and W. Jenkins. Redundant residue number systems for error detection and correction in digital filters. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(5):538 – 545, oct 1980. 25, 28

[EKD+03] D. Ernst, Nam Sung Kim, S. Das, S. Pant, R. Rao, Toan Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. Razor: a low-power pipeline based on circuit-level timing speculation. In *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, pages 7 – 18, dec. 2003. 6

[ENW08] F.L. Kastensmidt E.H. Neto and G. Wirth. Tbulk-bics: A built-in current sensor robust to process and temperature variations for soft error detection. *IEEE Transactions on Nulcear Science*, 55:2281–2288, 2008. 5, 113

[fS] International Technology Roadmap for Semiconductors 2011. Itrs home page. `http://www.cadence.com/`. 7, 113

[Gal63] R. Gallager. *Low Density Parity Check Codes*. MIT Press, 1963. 52, 54, 56, 57, 60, 67, 70, 121

[Gar59] Barvey L. Garner. The residue number system. *Managing Requirements Knowledge, International Workshop on*, 0:146, 1959. 10, 20, 45, 114

[Gil61] Arthur Gill. Cascaded finite-state machines. *Electronic Computers, IRE Transactions on*, EC-10(3):366 –370, sept. 1961. 86

[GR03] V.C. Gaudet and A Rapley. Iterative decoding using stochastic computation. *Electronics Letters*, 39(3):299–301, 2003. 7, 13, 51, 52, 62, 63, 77, 94

[GSN+07] M.J. Gadlage, R.D. Schrimpf, B. Narasimham, B.L. Bhuva, P.H. Eaton, and J.M. Benedetto. Effect of voltage fluctuations on the single event transient response of deep submicron digital circuits. *Nuclear Science, IEEE Transactions on*, 54(6):2495 –2499, dec. 2007. 37

[HAM50] R. W. HAMMING. Error detecting and error correcting codes. *BELL SYSTEM TECHNICAL JOURNAL*, 29(2):147–160, 1950. 121, 122

[JA88] W.K. Jenkins and E.J. Altman. Self-checking properties of residue number error checkers based on mixed radix conversion. *Circuits and Systems, IEEE Transactions on*, 35(2):159 –167, feb 1988. 25, 28

[Jen83] W.K. Jenkins. The design of error checkers for self-checking residue number arithmetic. *Computers, IEEE Transactions on*, C-32(4):388 –396, april 1983. 25, 28

[JM05a] Wonjin Jang and A.J. Martin. SEU-tolerant QDI circuits. *Asynchronous Circuits and Systems, 2005. ASYNC 2005. Proceedings. 11th IEEE International Symposium on*, pages 156–165, March 2005. 7

[JM05b] Wonjin Jang and Alain J. Martin. Soft-error robustness in QDI circuits. In *Proc. 1st Workshop on System Effects of Logic Soft Errors*, April 2005. 7

[Kat96] R.S. Katti. A new residue arithmetic error correction scheme. *Computers, IEEE Transactions on*, 45(1):13 –19, jan 1996. 26

[KFL01] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001. 54, 59, 63, 64, 77, 122

[KH04] T. Karnik and P. Hazucha. Characterization of soft errors caused by single event upsets in cmos processes. *Dependable and Secure Computing, IEEE Transactions on*, 1(2):128 – 143, april-june 2004. 6

[KLS92] H. Krishna, K.-Y. Lin, and J.-D. Sun. A coding theory approach to error control in redundant residue number systems. i. theory and single error correction. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 39(1):8 –17, jan 1992. 24, 26

[KP74] S. Kamal and C.V. Page. Intermittent faults: A model and a detection procedure. *IEEE Transactions on Computers*, 23:713–719, 1974. 5, 113

[Lal85] P. K. Lala. *Fault Tolerant and Fault Testable Hardware Design*. Prentice Hall, Englewood Cliffs, N.J., 1985. 5

[LCAP08] C.A. Lisboa, L. Carro, C. Argyrides, and D.K. Pradhan. Algorithm level fault tolerance: A technique to cope with long duration transient faults in matrix multiplication algorithms. In *VLSI Test Symposium, 2008. VTS 2008. 26th IEEE*, pages 363 –370, 27 2008-may 1 2008. 6

[LCB+10] Larkhoon Leem, Hyungmin Cho, Jason Bau, Quinn A. Jacobson, and Subhasish Mitra. Ersa: error resilient system architecture for probabilistic applications. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '10, pages 1560–1565, 2010. 5, 6, 15, 16, 20

[LCM84] Shu Lin, D. Costello, and M. Miller. Automatic-repeat-request error-control schemes. *Communications Magazine, IEEE*, 22(12):5 –17, december 1984. 42

[LDJK94] P. Liden, P. Dahlgren, R. Johansson, and J. Karlsson. On latching probability of particle induced transients in combinational networks. In *Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on*, pages 340 –349, jun 1994. 6

[LPG12] F. Leduc-Primeau and W. J. Gross. Faulty gallager-b decoding with optimal message repetition. In *Proceedings of 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton 2012)*, October 2012. 70

[LV62] R. E. Lyons and W. Vanderkulk. The use of triple-modular redundancy to improve computer reliability. *IBM J. Res. Dev.*, 6:200–209, 1962. 10, 16, 40, 115

[Mac03] David MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003). 10

[Man72] David Mandelbaum. Error correction in residue arithmetic. *Computers, IEEE Transactions on*, C-21(6):538 –545, june 1972. 21, 25

[MB59] D.E. Muller and W.S. Bertky. A theory of asynchronous circuits. In *Proc. International Symposium on the Theory of Switching, Part 1*, pages 204–243, 1959. 7, 70

[McE02] R. J. McEliece. *The Theory of Information and Coding*. Cambridge Univ. Press, 2002. 22

[Mes82] G. C. Messenger. Collection of charge on junction nodes from ion tracks. *Nuclear Science, IEEE Transactions on*, 29(6):2024 –2031, dec. 1982. 121

[MKW08] Konstantin Moiseev, Avinoam Kolodny, and Shmuel Wimer. Timing-aware power-optimal ordering of signals. *ACM Trans. Des. Autom. Electron. Syst.*, 13(4):65:1–65:17, October 2008. 109

[Moo05] T. K. Moon. *Error Correction Coding*. New Jersey: John Wiley and Sons, 2005. 121

[Mul07] O. Muller. *Generic multiprocessor architectures for high-throughput turbo-communications*. h.D. dissertation, Univ. Bretagne-Sud and TELECOM Bretagne, France, 2007. 39

[MZM10] Natasa Miskov-Zivanov and Diana Marculescu. Formal modeling and reasoning for reliability analysis. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 531 –536, june 2010. 36

[MZS+07] S. Mitra, Ming Zhang, N. Seifert, T.M. Mak, and Kee Sup Kim. Built-in soft error resilience for robust system design. In *Integrated Circuit Design and Technology, 2007. ICICDT '07. IEEE International Conference on*, pages 1 –6, 30 2007-june 1 2007. 7, 72

[NBS+07] B. Narasimham, B.L. Bhuva, R.D. Schrimpf, L.W. Massengill, M.J. Gadlage, O.A. Amusan, W.T. Holman, A.F. Witulski, W.H. Robinson, J.D. Black, J.M. Benedetto, and P.H. Eaton. Characterization of digital single event transient pulse-widths in 130-nm and 90-nm cmos technologies. *Nuclear Science, IEEE Transactions on*, 54(6):2506 –2511, dec. 2007. 37

[Nea] Radford Neal. Software for low density parity check (ldpc) codes. `http://www.cs.toronto.edu/~radford/ldpc.software.html`. 126

[OP07] A. Omandi and B. Premkumar. *Residue Number Systems: Theory and Implementation*. World Scientific Book, 2007. 21

[Ram83] V. Ramachandran. Single residue error correction in residue number systems. *Computers, IEEE Transactions on*, C-32(5):504 –507, may 1983. 26

[RM00] S.K. Reinhardt and S.S. Mukherjee. Transient fault detection via simultaneous multithreading. In *Computer Architecture, 2000. Proceedings of the 27th International Symposium on*, pages 25 –36, june 2000. 5

[RMR08] A. Marelli R. Micheloni and R. Ravasio. *Error correction codes for non-volatile memories*. Berlin, Germany: Springer-Verlag, 2008. 5

[ROK09] Wenjing Rao, A. Orailoglu, and R. Karri. Logic mapping in crossbar-based nanoarchitectures. *Design Test of Computers, IEEE*, 26(1):68 –77, jan.-feb. 2009. 64, 65, 116

[RU01] T. J. Richardson and R. L. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 42(2):599–618, 2001. 14, 55, 56, 59, 60, 112

[Sah91] C.-T. Sah. *Fundamentals of solid-state electronics*. World Scientific, 1991. 109

[SAKJ10] Naresh R. Shanbhag, Rami A. Abdallah, Rakesh Kumar, and Douglas L. Jones. Stochastic computation. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 859 –864, june 2010. 5

[SH04] J. Sequra and C. F. Hawkins. *MOS Electronics: How it works, how it fails*. Wiley-IEEE Press, 2004. 5, 113

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell Systems Technology*, 27:379–423, 623–656, 1948. 22, 53

[SK89] A.P. Shenoy and R. Kumaresan. Fast base extension using a redundant modulus in rns. *Computers, IEEE Transactions on*, 38(2):292 –297, feb 1989. 22

[SK92] J.-D. Sun and H. Krishna. A coding theory approach to error control in redundant residue number systems. ii. multiple error detection and correction. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 39(1):18 –34, jan 1992. 24, 26

[SKHW05] G. Snider, P. Kuekes, T. Hogg, and R. Stanley Williams. Nanoelectronic architectures. *Applied Physics A: Materials Science & Processing*, 80:1183–1195, 2005. 10.1007/s00339-004-3154-4. 86

[SL90] C.-C. Su and H.-Y. Lo. An algorithm for scaling and single residue error correction in residue number systems. *Computers, IEEE Transactions on*, 39(8):1053 –1064, aug 1990. 26

[Spe] Cadence Virtuoso Spectre. Cadence home page. `http://public.itrs.net/`. 82

[Sri96] G. R. Srinivasan. Modeling the cosmic-ray-induced soft-error rate in integrated circuits: An overview. *IBM Journal of Research and Development*, 40(1):77 –89, jan. 1996. 121

[SS06] B. Shim and N.R. Shanbhag. Energy-efficient soft error-tolerant digital signal processing. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(4):336 –348, april 2006. 6, 15, 16, 18, 40

[SSS04] Byonghyo Shim, Srinivasa R. Sridhara, and Naresh R. Shanbhag. Reliable low-power digital signal processing via reduced precision redundancy. *IEEE Trans. VLSI Syst.*, 12(5):497–510, 2004. 6, 15, 16, 17

[ST67] N. S. Szabo and R. I. Tanaka. *Residue Arithmetic and its Applications to computer Technology*. New York: McGraw-Hill, 1967. 10, 20, 21, 114

[SYH+06] T. Suzuki, Y. Yamagami, I. Hatanaka, A. Shibayama, H. Akamatsu, and H. Yamauchi. A sub-0.5-v operating embedded sram featuring a multi-bit-error-immune hidden-ecc scheme. *Solid-State Circuits, IEEE Journal of*, 41(1):152 – 160, jan. 2006. 5

[Tay84] F.J. Taylor. Residue arithmetic a tutorial with examples. *Computer*, 17(5):50 –62, may 1984. 21, 26

[TBJJ10] Yangyang Tang, Emmanuel Boutillon, Christophe Jégo, and Michel Jézéquel. A new single-error correction scheme based on self-diagnosis residue number arithmetic. In *DASIP*, pages 27–33, 2010. 6, 10, 15, 33, 45, 115

[TBJJ11] Yangyang Tang, E. Boutillon, C. Jego, and M. Jezequel. Hardware efficiency versus error probability in unreliable computation. In *Signal*

Processing Systems (SiPS), 2011 IEEE Workshop on, pages 168 –173, oct. 2011. 17, 115

[TBW+13] Yangyang Tang, Emmanuel Boutillon, Chris Winstead, Christophe Jégo, and Michel Jézéquel. Muller c-element based decoder (mcd): A decoder against transient faults. In *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, may 2013. 63, 65, 82, 110

[TSW+12] Yangyang Tang, Gopalakrishnan Sundararajan, Chris Winstead, Emmanuel Boutillon, Christophe Jégo, and Michel Jézéquel. Techniques and prospects for fault-tolerance in post-cmos ulsi. In *21st International Workshop on Post-Binary ULSI Systems, ULSIWS*, 2012. 52, 77

[TWB+12] Yangyang Tang, Chris Winstead, Emmanuel Boutillon, Christophe Jégo, and Michel Jézéquel. An ldpc decoding method for fault-tolerant digital logic. In *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, pages 3025 –3028, may 2012. 7, 52, 63, 70, 77

[Ung82] G. Ungerboeck. Channel coding with multilevel/phase signals. *Information Theory, IEEE Transactions on*, 28(1):55 – 67, jan 1982. 14, 99, 101, 103, 112

[Var11] Lav R. Varshney. Performance of ldpc codes under faulty iterative decoding. *IEEE Transactions on Information Theory*, 57(7):4427–4444, 2011. 22, 73, 99, 101

[VC07] B. Vasic and S.K. Chilappagari. An information theoretical framework for analysis and design of nanoscale fault-tolerant memories based on low-density parity-check codes. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 54(11):2438 –2446, nov. 2007. 70

[vN55] J. von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In C. E. Shannon, editor, *Automata Studies*, pages 43–98. Princeton University Press, Princeton, NJ, 1955. 22

[WEH09] C. Winstead and M. El Hamoui. Reducing clock jitter by using muller-c elements. *Electronics Letters*, 45(3):150 –151, 29 2009. 7, 70

[WGRS05] C. Winstead, V. C. Gaudet, A. Rapley, and C. Schlegel. Stocahstic iterative decoders. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, 2005. 7, 13, 51, 52, 62, 63, 70, 77, 94

[WH66] R.W. Watson and C.W. Hastings. Self-checked computation using residue arithmetic. *Proceedings of the IEEE*, 54(12):1920 – 1931, dec. 1966. 20, 22

[WH09] C. Winstead and S. Howard. Probabilistic LDPC-coded fault compensation technique for reliable nanoscale computing. *IEEE Transactions on Circuits and Systems II – Express Briefs*, 56(6):484–488, June 2009. 7, 12, 15, 51, 52, 62, 63, 67, 70, 85, 115

[Win09] C. Winstead. C-element multiplexing for fault-tolerant logic circuits. *Electronics Letters*, 45(19):969 –970, 10 2009. 7, 70

[WLMT11] Chris Winstead, Yi Luo, Eduardo Monzon, and Abiezer Tejeda. An error correction method for binary and multiple-valued logic. *Multiple-Valued Logic, IEEE International Symposium on*, 0:105–110, 2011. 7, 15, 70, 71, 72, 86, 111

[WTB+12] Chris Winstead, Yangyang Tang, Emmanuel Boutillon, Christophe Jego, and Michel Jezequel. A space-time redundancy technique for embedded stochastic error correction. In *Turbo Codes and Iterative Information Processing (ISTC), 2012 7th International Symposium on*, pages 36 –40, aug. 2012. 7, 63, 77

[YH01] Lie-Liang Yang and L. Hanzo. Redundant residue number system based error correction codes. In *Vehicular Technology Conference, 2001. VTC 2001 Fall. IEEE VTS 54th*, volume 3, pages 1472 –1476 vol.3, 2001. 22

[YL73] S.S.-S. Yau and Yu-Cheng Liu. Error correction in redundant residue number systems. *Computers, IEEE Transactions on*, C-22(1):5 – 11, jan. 1973. 24, 25

[ZMM+06a] M. Zhang, S. Mitra, T. M. Mak, N. Seifert, N. J. Wang, Q. Shi, K. S. Kim, N. R. Shanbhag, and S. J. Patel. Sequential element design with built-in soft error resilience. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(12):1368 –1378, 2006. 7, 70

[ZMM+06b] Ming Zhang, Subhasish Mitra, T. M. Mak, Norbert Seifert, Nicholas J. Wang, Quan Shi, Kee Sup Kim, Naresh R. Shanbhag, and Sanjay J. Patel. Sequential element design with built-in soft error resilience. *IEEE Trans. VLSI Syst.*, pages 1368–1378, 2006. 7

List of figures

List of tables